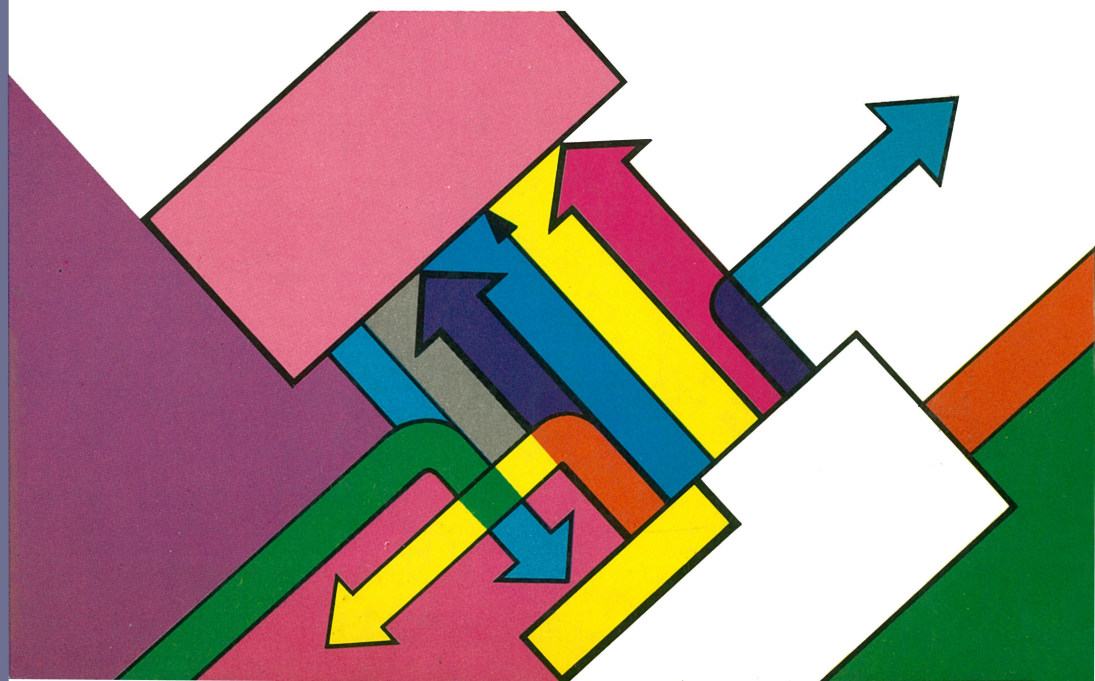
il BUGBOOK® III

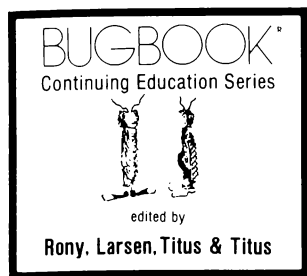
INTERFACCIAMENTO E PROGRAMMAZIONE DEL MICROCOMPUTER 8080

EDIZIONE ITALIANA

Peter R. Rony
David G. Larsen
Jonathan A. Titus

JACKSON ITALIANA EDITRICE

Il **dr. Peter R. Rony** è professore presso il Dipartimento di Ingegneria Chimica del Virginia Polytechnic Institute & State University (USA). L'elettronica digitale ed i microcomputer giocheranno un ruolo molto importante nei controlli di processo, soggetto questo di considerevole interesse per l'ingegneria chimica. Il dr. Rony è coautore di molti altri Bugbooks e di una pubblicazione mensile, denominata Columns, riguardante l'interfacciamento dei microcomputer, che appare su *American Laboratory, Computer Design, Ham Radio Magazine,* in Usa, *Electroniker* in Germania e *Elettronica Oggi* in Italia.

Mr. David G. Larsen è un istruttore del Dipartimento di Ingegneria Chimica del Virginia Polytechnic Institute & State University dove svolge attività didattica a vari livelli nel campo dell'elettronica analogica e digitale. È coautore di altri Bugbook e di una pubblicazione mensile, denominata Columns, riguardante l'interfacciamento dei microcomputer. Con il dr. Rony cura una serie di corsi sui microcomputer, sotto gli auspici della Extension Division della suddetta Università. Questi corsi sono particolarmente apprezzati e seguiti da professionisti di ogni parte del mondo.

Mr. Jonathan A. Titus è presidente della Tychon Inc. La maggior parte della sua attività alla Tychon è concentrata sulle applicazioni di microcomputer e sui piccoli sistemi a microcomputer. Ha scritto ed ha collaborato alla realizzazione di articoli sulla strumentazione e sui microcomputer, pubblicati su riviste sia a carattere professionale che hobbistico. Jon iniziò ad interessarsi ai microcomputer nel '71 quando lavorava all'automazione di strumenti per analisi chimica. La sua prima esperienza fu il microcomputer MARK 8, che era basato sull'8008.. Nel luglio '74 venne presentato dalla rivista Radio Electronics come il primo hobbista di computer. Gli attuali interessi di Jon comprendono lo sviluppo di sitemi didattici, controllo digitale e sistemi di acquisizione dati a distanza.

il BUGBOOK® III

INTERFACCIAMENTO E PROGRAMMAZIONE DEL MICROCOMPUTER 8080

PETER R. RONY
Department of Chemical Engineering

DAVID G. LARSEN
Department of Chemistry

Virginia Polytechnic Institute & State University
Blacksburg, Virginia 24061

e

JONATHAN A. TITUS
Tychon, Inc.
Blacksburg, Virginia 24060

Versione italiana

ALDO CAVALCOLI
Mipro s.r.l.
Via Carducci, 15
20123 Milano

JACKSON
ITALIANA
EDITRICE

Piazzale Massari, 22
20125 Milano

Prefazione all'Edizione Italiana

La necessità di un libro di testo in italiano sul microprocessore 8080A era da tempo sentita.

Il settore dell'elettronica che gravita attorno ai microprocessori ed ai microcomputer si sviluppa in modo tanto rapido da rendere difficoltoso un costante ed adeguato aggiornamento.

Facendo un'approfondita indagine del mercato italiano e analizzando le attuali esigenze degli utenti di microprocessori, abbiamo individuato la necessità di una corretta, completa e dettagliata informazione sul microprocessore 8080A per questi motivi fondamentali:

1) L'8080A sta per diventare standard in molte aziende italiane sia per scelta polarizzata verso il prodotto più usato sia per imitazione di eventuali consociate estere.
2) L'8080A è alla base di tutto un filone di microprocessori «filosoficamente» equivalenti, e cioè l'8085, 8048 e derivati, l'8086 e ancora la serie Zilog, Z-80, Z-8, Z-8000. Fornire un buon libro sull'8080A, quindi, significa nozioni e strumenti di lavoro sul sistema base a microprocessori e di riflesso su molti dei più affermati microprocessori esistenti.

Questo libro inoltre, è stato giudicato negli Stati Uniti uno dei migliori testi sui microprocessori disponibili. Esso è nato dalla quotidiana sperimentazione didattica al Virginia Polytechnic Institute State University.

Anche in Italia la Mipro ha deciso di adottare questo volume come libro di testo nei suoi corsi, sia aziendali che di lunga durata, a partire dal secondo semestre del 1978.

Aldo Cavalcoli

Prefazione

Nel 1971 fecero la loro prima comparsa i microprocessori. Si trattò del microprocessore a 4 bit 4004 della INTEL, il quale, per quanto di non comoda utilizzazione, aprì comunque la strada all'impiego dei microprocessori nel campo delle applicazioni di controllo, settore in cui attualmente l'impiego è enorme. Seguì dopo breve tempo l'INTEL 8008, microprocessore ad 8 bit: questo microprocessore permise all'utente di realizzare piccoli computer «general purpose». Successivamente molte compagnie iniziarono ad offrire schede logiche standard con la necessaria logica di controllo, memoria e buffer di ingresso/uscita, con conseguente realistico utilizzo in applicazioni su larga scala.

Dopo l'annuncio dell'8008, molte ditte di semiconduttori, si dettero da fare per non perdere il treno dei microprocessori.

Seguì l'INTEL con il più potente 8080, la MOTOROLA con il 6800, la FAIRCHILD con l'F-8, ed altre dozzine di case.

Nel 1977 nasce una nuova linea: i «computer-on-a-chip», che contengono, in un singolo chip di circuito integrato, tutta la logica necessaria, la memoria ed i collegamenti di ingresso/uscita necessari alla realizzazione di un piccolo microcomputer. I microcomputer non solo stanno trovando un enorme successo nei controlli o nella strumentazione, ma anche nel mercato dell'«hobby computer».

Cinque anni fa, chi si occupava dei computer per hobby, poteva solo sognare di possedere un proprio calcolatore, ma ben pochi potevano permetterselo.

Oggi, migliaia di amatori, posseggono il loro piccolo computer, utilizzato per applicazioni di tipo generale, giochi, piccola gestione dell'economia familiare, gestioni di magazzini, sistemi di controllo energetico per uso domestico. La pubblicazione dell'articolo relativo alla costruzione del Mark 8, apparso sul numero di Luglio della rivista *«Radio-Electronics»*, dette il via ad una inarresta-

bile diffusione dell'argomento. Basato sul microprocessore 8080, il Mark 8 metteva a disposizione degli appassionati di computer, i mezzi per avere il proprio calcolatore personale.

Fu seguito poi da versioni commerciali, costruite dalla società MITS, dalla South West Technical Products Corp. e da altre case; attualmente migliaia di microcomputer si trovano nelle case americane. Questo è appena l'inizio. Possiamo tranquillamente affermare che centri domestici di passatempo e comunicazioni basati sul connubio televisione, chip LSI per giochi e microcomputer, saranno venduti a milioni, come il maggior prodotto «consumer» dei prossimi anni.

Se gli organi governativi avranno un corretto approccio alla riinvenzione delle comunicazioni, causata dalla tecnologia LSI e dalle fibre ottiche, il prossimo passo sarà quello di approntare i sistemi di comunicazione digitali (tra le case, gli uffici, centri servizi vari, negozi) che microcomputer a basso costo porteranno senz'altro in ogni settore della vita.

Già ora li troviamo nelle automobili, nelle macchine da cucire, nei forni a microonde, nei giochi televisivi, nei punti di vendita, nelle stazioni di servizio per carburanti, nei tassametri, e così via. Domani saranno nei telefoni, nelle lavatrici, nelle macchine da scrivere, nei sistemi di riscaldamento e di condizionamento, nei giocattoli...

Nasceranno nuove industrie, alcune già esistenti scompariranno: un minuto di silenzio per chi farà il grosso scivolone? Obiettivo di questo libro è insegnarvi i quattro punti fondamentali dell'interfacciamento dei microcomputer:

- Generazione degli impulsi di selezione dispositivo
- Uscite dai microcomputer
- Ingressi nei microcomputer
- Gestione interrupt

tutto nell'ambito di microcomputer basati sull'8080. Speriamo di potervi fornire i concetti base relativi all'interfacciamento dei microcomputer e le associate tecniche software di I/O, in modo da permettervi di sviluppare le vostre specifiche interfacce verso altri dispositivi, tipo teletype, display, pannelli di controllo, convertitori A/D e D/A, strumentazione da laboratorio ed altro.

Il Capitolo 1 discute il futuro ruolo dei microcomputer, i quattro punti base relativi all'interfacciamento e fa un riassunto dei concetti di elettronica digitale che vi saranno necessari nel proseguimento della lettura di questo libro.

Il Capitolo 2 descrive un piccolo microcomputer basato sul microprocessore 8080 e discute il concetto di microcomputer dal punto di vista dell'I/O in dettaglio, soprattutto per quanto riguarda l'uso degli impulsi di selezione dispositivo per abilitare le operazioni dei circuiti integrati collegati al microprocessore.

Il Capitolo 3 è un'introduzione al software del microprocessore

8080. Tutto il set di istruzione è fornito in dettaglio. Si fa poi particolare attenzione alle operazioni di trasferimento dati, alle operazioni aritmetiche e logiche, istruzioni di branch, di ingresso-uscita, e decodifica dei registri.

Il Capitolo 4 tratta l'argomento molto importante della generazione degli impulsi di selezione dispositivo. Viene presentata una certa varietà di circuiti di decodifica sperimentali, atti a generare da 1 a 256 differenti impulsi di selezione dispositivo.

Il Capitolo 5 descrive una tecnica molto utile per attuare il conteggio del numero di cicli di clock richiesti per l'esecuzione di una sezione di un programma. La tecnica permette di verificare il numero di cicli teorici per ogni istruzione. Saranno presentati dei programmi standard in grado di generare ritardi di tempo multipli di 0,5 ms o 0,2 s.

Il Capitolo 6, forse il più difficile del libro, fornisce la descrizione delle operazioni interne del microprocessore 8080. Le temporizzazioni dell'8080 sono discusse in termini di stati, cicli macchina, cicli istruzione. Sono mostrati dei circuiti per dimostrare la generazione dei segnali $\overline{\text{IN}}$ e $\overline{\text{OUT}}$.

Il Capitolo 7 discute l'ingresso-uscita da un microcomputer, il latch dei dati in uscita, la tecnica del buffer three-state. Anche in questo caso si attua una esemplificazione dei concetti esposti tramite circuiti e programmi, al fine anche di insegnarvi come scegliere uno specifico tasto di una tastiera ASCII, come realizzare l'uscita dei dati verso un display, come acquisire dati digitali, e così via.

Il Capitolo 8 discute dettagliatamente i modi operativi dei microcomputer, le operazioni di polling, gli interrupt, le subroutine, i flag esterni, il mascheramento, lo stack, le strategie di gestione interrupt e le priorità di interruzione. Alla fine del testo, in appendice, è dato l'intero set di istruzioni dell'8080.

Fondamentalmente questo testo è una versione ampliata del Bugbook III: Microcomputer Interfacing Experiments Using the Mark 80 Microcomputer, an 8080 System (non edito in Italia, N.D.T.) venduto dalla E-L Instruments, Inc.; Derby, Connecticut.

Gli esperimenti sono stati riscritti ed aggiornati. Il capitolo degli interrupt e dei flag è considerevolmente espanso. Sono state apportate molte modifiche dal punto di vista editoriale, per rendere più chiaro il testo.

In questo libro, si ipotizza che il lettore abbia una conoscenza di base dei concetti dell'elettronica digitale, cioè delle porte logiche, del flip-flop, dei decoder, multiplexer, buffer three-state, ecc.

Questi concetti sono stati presentati nei libri *Bugbook I, II - Esperimenti sui Circuiti Logici e di Memoria tramite l'Utilizzo di Circuiti Integrati TTL* (editi nella versione italiana dalla Jackson Italiana, Editrice). Se poi il lettore desidera interfacciare un microcomputer con una teletype, od un display, sono necessari i concetti relativi al trasferimento dati asincrono seriale. Tali concetti sono presentati nel

Bugbook II A; Esperimenti di Interfacciamento e Comunicazione Dati tramite l'utilizzo del Ricevitore/Trasmettitore Universale Asincrono (UART) e del Loop di Corrente di 20 mA (edito nella versione italiana dalla Jackson Italiana Editrice).

Gli autori di questo libro ed il Dr. Cristopher A. Tytus, che ha collaborato alla stesura, sono coinvolti quotidianamente nella didattica sui microprocessori e portano avanti una personale attività di progettazione. Brevi corsi sull'elettronica digitale e sull'interfacciamento dei microcomputer sono realizzati dalla Tychon, Inc. [Tel. (703) 951-9030] e dal Continuing Education Center e Extention Division del Virginia Polytechnic Institute e State University [Dr. Norris Bell, tel. (703) 951-6328]. In Italia la formazione nel campo dei microprocessori è portata avanti dalla società MIPRO, sia per i corsi aziendali che per i corsi di lunga durata. Per informazioni contattate direttamente la Sezione Didattica della MIPRO [Ing. Cavalcoli, oppure Sig.ra Cavenaghi Tel. (02) 879062].

Se siete interessati ad un aggiornamento sulle novità in campo editoriale, richiedete il catalogo dei libri della Jackson Italiana Editrice (Piazzale Massari, 22 - 20125 Milano).

Un ringraziamento, per il continuo sostegno, a Mr. Murray Gallant ed alla E-L Instruments, Inc., sempre vicini ai nostri sforzi didattici nel campo dell'elettronica.

Peter R. Rony
David G. Larsen
Jonathan A. Tytus

Indice

CAPITOLO 1

CAPITOLO 2

CAPITOLO 3

CAPITOLO 4

CAPITOLO 5

CAPITOLO 6

CAPITOLO 7

CAPITOLO 8

APPENDICE 1

APPENDICE 2

APPENDICE 3

IL GRUPPO DI BLACKSBURG

I circuiti integrati a larga scala o "chips" LSI stanno creando una seconda rivoluzione industriale che ben presto ci coinvolgerà tutti. La velocità degli sviluppi in questo settore è enorme e diviene sempre più difficile stare al passo coi progressi che si stanno compiendo.

E' sempre stato nostro obiettivo, come Gruppo di Blacksburg, creare tempestivamente e concretamente materiali didattici ed aiuti tali da permettere a studenti, ingegneri, tecnici, ecc. di sfruttare le nuove tecnologie per le loro esigenze particolari. Stiamo facendo questo in molti modi, con libri di testo, brevi corsi, articoli mensili di "computer interfacing" e attraverso la creazione di "hardware" didattico.

I membri del nostro gruppo hanno creato la loro sede a Blacksburg, fra le montagna Appalache del sud-ovest Virginia. Mentre era in corso di preparazione attiva la nostra collaborazione di gruppo, i membri si sono occupati di elettronica digitale, minicomputer e microcomputer.

I nostri sforzi in Italia nel campo didattico sono stati:
- Introduzione, avvenuta nel 1976, sul mercato Italiano da parte della Microlem s.p.a. (Milano) della nostra linea di moduli basati sul sistema di breadboarding senza saldature, o moduli OUTBOARDS®, che facilitano la progettazione e la prova dei circuiti digitali rispetto ai sistemi tradizionali.

- Traduzione e pubblicazione da parte della JACKSON ITALIANA EDITRICE s.r.l., iniziata nel 1978, dei BUGBOOKS® e della collana di libri Blacksburg Continuing Education Series® che comprende una ventina di titoli riguardanti: l'elettronica di base, microcomputer, convertitori analogico/digitali e digitali/analogici, software per microcomputer, amplificatori operazionali, filtri attivi, phase-locked loops ecc. In ogni libro, oltre al normale testo, vi sono esempi ed esperimenti condotti col sistema passo-passo. Noi crediamo che la sperimentazione consenta di rafforzare i concetti base. Molti titoli stanno per essere tradotti oltre che in Italiano, anche in Spagnolo, Tedesco, Giapponese e Cinese.

- Organizzazione da parte della Microlem, in collaborazione col Virginia Polytechnic Institute and State University e la MIPRO, inizia nel dicembre 1977, di brevi corsi sull'elettronica digitale, la programmazione e l'interfacciamento dei microcomputer. Per l'intera durata dei corsi i partecipanti utilizzano i moduli OUTBOARDS e il microcomputer MMD-1 per verificare i concetti di elettronica digitale, interfacciamento e programmazione presentati nei Bugbooks V e VI. Gli interessati a questi corsi possono rivolgersi alla segreteria dei "Blacksburg Continuing Courses in Italy" tel. (02) 27 10 465.

- Pubblicazione da parte della JACKSON ITALIANA EDITRICE, inziata nel 1978, di articoli, denominati Column, su "Microcomputer Interfacing" nella qualificatissima rivista ELETTRONICA OGGI. Questi columns appaiono anche in quattro riviste americane e in altre tre riviste di elettronica delle quali una Australiana, una Svizzera e una Sud Africana, raggiungendo circa 1.500.000 lettori ogni mese.

- Collaborazione con la SGS-ATES, iniziata nel 1978, per la stesura di materiale didattico relativo alla programmazione e all'interfacciamento del microcomputer SGS-ATES single-board Z-80. Oltre a ciò siamo stati in grado di fornire un prodotto integrato: prodotto progettato per un materiale didattico e contemporaneamente materiale didattico progettato per il prodotto.

- Introduzione da parte dei membri del gruppo di Blacksburg di tecniche didattiche che includono l'uso combinato di stazioni sperimentali multipersone, testi per uso di laboratorio e diapositive 35 mm relative ai testi.
Tutto ciò è stato definito, da alcuni insegnanti italiani, come "il nuovo sistema per la didattica italiana".

Mr. David Larsen e il dr. Peter Rony fanno parte della facoltà dei dipartimenti di Chimica e Ingegneria Chimica del Virginia Polytechnic Institute & State University. Mr. Jonathan Titus e il dr. Christopher Titus fanno parte della Tychon Inc. tutti di Blacksburg, Virginia.

CAPITOLO 1

Cosa è un Microcomputer?

In questo libro, studierete alcuni casi reali tesi a dimostrare i principi, i concetti e le applicazioni di un *microcomputer* ad 8 bit, basato sul *microprocessore 8080.* Nel fare questo, di fatto parteciperete alla nuova «rivoluzione» nel campo dell'elettronica, che trasformerà il *computer* da macchina di grandi dimensioni, costosa e di non facile utilizzo, in un dispositivo compatto, a basso costo, utilizzabile su larga scala per gli usi più disparati: tra non molto troverete i computer dappertutto, nella vostra automobile, in casa, nei luoghi più impensati. Questi piccoli computer avranno una larghissima diffusione ed una profonda influenza in qualunque settore della vita di tutti i giorni.

I grandi sistemi di elaborazione dati continueranno a realizzare complicate operazioni matematiche, mentre i semplici calcolatori tascabili attueranno i calcoli più semplici. Dato che l'aspetto più comune dei calcolatori è quello del calcolo, ci si può chiedere: dove si collocano i microcomputer? Dopotutto, se i grandi calcolatori fanno tutti i calcoli complessi e se le calcolatrici tascabili fanno quelli più semplici, resta qualcosa per i microcomputer? La risposta nasce dalla seguente considerazione: i calcolatori hanno due importanti funzioni:

1 - *Number crunching,* cioè macchina i cui dati in forma digitale sono manipolati ad alta velocità.

2 - *Controllore digitale programmabile,* per il controllo di macchine tramite l'invio o la ricezione di segnali digitali.

L'uso principale dei microcomputer sarà come controllori, non come number-crunching.

Potreste avere un'idea del potenziale mercato dei microprocessori se solo foste a conoscenza del numero delle macchine che potrebbero essere controllate da sistemi basati su microcomputer. Il telefono può usare il microprocessore, come pure una macchina da scrivere elettrica, un televisore, un giocattolo sofisticato, un impianto stereo, una lavatrice, e tanti altri dispositivi della vita quotidiana. In ogni casa, ufficio, industria o laboratorio, ci sono circa da 3 a 10 macchine automatizzabili con microprocessori. Nel caso dell'Italia, con 60 milioni di abitanti, questo vuol dire qualcosa come 200 milioni, se non 1 miliardo, di potenziali microcomputer.

In realtà ad esempio, negli USA si parla per ora di 100 milioni di microcomputer, il che è già una gran cosa.

L'argomento è vasto, e tutta una serie di testi può essere scritta per descrivere l'architettura dei calcolatori, il loro funzionamento e le loro applicazioni. Questi non sono gli scopi di questo libro, che differisce da testi già scritti in quanto si vuole dare enfasi all'uso dei microcomputer soprattutto come controllori.

Imparerete come *interfacciare* un microcomputer; la parola «interfacciare» significa realizzare un collegamento tra microcomputer ed un dispositivo esterno, in modo tale che i due lavorino in un modo coordinato. ([1]*) Verrete sensibilizzati solo su un microcomputer realizzato attorno al microprocessore 8080, per questi motivi:

1 - Oramai sono parecchi i costruttori dell'8080, le cosiddette «seconde sorgenti», per cui è presumibile che microcomputer basati sull'8080 diventeranno sempre più comuni.
2 - Dalla fine del 1977, l'8080 è venduto a circa 10 $ per singolo pezzo, da cui una notevole diffusione di questo dispositivo (ed il prezzo continua a scendere).
3 - Il set di istruzioni dell'8080 è abbastanza potente ed i programmi sono sufficientemente semplici da scriversi.
4 - L'8080 è relativamente veloce: un'istruzione aritmetica, come una somma o sottrazione, è realizzabile in 2 μs, e si prevedono versioni più veloci dell'8080.
5 - Il range di indirizzamento dell'8080 è 65536 differenti locazioni di memoria ad 8 bit; può poi generare 256 diversi segnali di ingresso ed uscita.
6 - In base alla popolarità dell'8080, sarà disponibile una larga varietà di programmi.

Gli autori pensano che utilizzerete bene il vostro tempo se studierete i principi, i concetti e le applicazioni di un computer del futuro, come l'8080, piuttosto che un computer del passato, come può essere il PDP-8.

([1]*) Vedere l'appendice 1 per i riferimenti indicati nel testo con un asterisco (*).

Questo libro si riferisce a 4 punti fondamentali dell'interfacciamento:

1 - *Generazione di segnali di selezione dispositivo.*
2 - *Latch di dati in uscita.*
3 - *Acquisizione di dati in ingresso.*
4 - *Gestione delle interruzioni.*

Un'ampia serie di esempi vi sarà fornita unitamente ai concetti ed alle tecniche necessarie per lo sviluppo di vostri circuiti e di vostri programmi finalizzati alla realizzazione di uno o più dei punti elencati. Una volta realizzate queste funzioni base, avrete sotto controllo il mondo dei microcomputer.

INTRODUZIONE A QUESTO CAPITOLO

In questo capitolo, incontrerete una serie di esempi che vi aiuteranno ad acquisire competenza nell'interfacciamento dei microcomputer. Prima di fare ciò, sarà senz'altro utile capire cosa è un computer e quale è la distinzione tra *microcomputer, minicomputer, computer, controllore, processore di dati e processore logico.*

In più riteniamo importante fare un'attenta indagine delle caratteristiche del «bug» principale di questo libro, il microprocessore 8080. Questo chip, realizzato nel 1974 dalla Intel Corporation, è una completa unità centrale ad 8 bit (CPU), detta anche MPU (Micro-Processor-Unit), fornita in un singolo chip LSI. Semplici istruzioni possono essere eseguite in 2 μs; tenete presente che la velocità del famoso PDP-8/E della Digital Equipement Corporation è di 1,2 μs. L'8080 ha come «seconda sorgente» la Texas Instruments, la National Semiconductor, la NEC, la Siemens, la Hitachi; da questo si desume che le industrie vedono nell'8080 una forza importante nel mercato dei microprocessori.

La discussione sull'8080 sarà suddivisa nei seguenti argomenti:

- Configurazione dei pin e funzioni dei pin dell'8080.
- Organizzazione di un tipico microcomputer che usa l'8080.
- Operazioni interne dell'8080.
- Set di istruzioni dell'8080.

Partendo da questo capitolo e continuando in altri, vedremo ciascuno di questi argomenti. Il dettaglio con cui saranno trattati gli argomenti, varierà a seconda dei casi. L'obiettivo non è certo quello di bombardarvi con una serie di diagrammi di tempo, *cicli macchina,* quanto piuttosto aiutarvi a sviluppare la vostra esperienza. Per i lettori interessati ad avere ulteriori informazioni sul microprocessore 8080 gli autori raccomandano le seguenti fonti:

— Robert H. Cushman, «The Intel 8080: First of the second-gene-

ration microprocessors», *Electronic Design News* 19 (9), p. 30 (5 Maggio, 1974).
— Masatoshi Shima and Federico Faggin, «In switch to n-MOS, microprocessor gets a 2-μs cycle time», *Electronics* 47 (8), p. 95 (18 Aprile, 1974).
— Intel Corp., *Intel Intellec 8/Mod 80 Microcomputer Development System Reference Manual,* Santa Clara, California, 1975.
— Intel Corp., *Intel 8080 Microcomputers System Manual,* Santa Clara, California, Settembre 1976.
— Adam Osborne, *An Introduction to Microcomputers, Volume II. Some Real Products.* Osborne Associates, Berkeley, California, 1976.
— Adam Osborne, *8080 Programming for Logic Design,* Osborne Associates, Berkeley, California, 1976.
— W. J. Weller, A. V. Shatzel, and H. Y. Nice, *Pratical Microcomputer Programming. The Intel 8080,* Northern Technology Books, 1976.
— A. Cavalcoli, V. Scibilia, *Introduzione al microprocessore 8080,* dispense dei corsi MIPRO, Milano, 1977.

OBIETTIVI

Alla fine di questo capitolo, sarete in grado di:

- Spiegare la differenza tra microcomputer e microprocessore.
- Definire i seguenti termini: computer, computer digitale, data processor, controllore, hardware, software, memoria, parola di memoria, indirizzo di memoria, dato di memoria, lettura, scrittura, accesso casuale, memoria a sola lettura, interfacciamento, impulso di selezione dispositivo, interrupt.
- Descrivere i diversi tipi di computer e controllore sulla base delle seguenti caratteristiche: lunghezza della parola, complessità, applicazione, costi, dimensione della memoria, programma, velocità, I/O, architettura, progetto da fare, volume di produzione.
- Descrivere le operazioni di un microcomputer basato sull'8080.

MICROPROCESSORI COME MICROCONTROLLORI E PROCESSORI LOGICI

Le applicazioni dei microprocessori tendono a cadere nelle seguenti categorie:

- Controllori
- Prodotti di consumo
- Comunicazioni

Tabella 1-1. Spettro della complessità dei sistemi elettronici, dalla semplice logica cablata ai sistemi di elaborazione dati ad elevata complessità. Questa tabella si basa su materiale della Log. Corp. per un articolo di Wallace B. Riley. 17 ottobre 1974 Electronics.

LUNGHEZZA DELLA PAROLA	1	2	4	8	16	32	64
COMPLESSITA'	Logica cablata	Array logici programmati	Calcolatrice	Microprocessor	Minicomputer	Grande computer	
APPLICAZIONE		Controllo		Calcoli Elaborazione dedicati		Dati generali a basso costo	Alte prestazioni nella elaborazione dati generali
COSTO	100$ (1974)		1000$ (1974)			10.000$ (1974)	100.000$ e più (1974)
GRANDEZZA DELLA MEMORIA	Molto piccola 0-4 parole	Piccola 2-10 parole		Media 10-1000 parole		Grande 1000-1 milione di parole	Molto grande più di 1 milione di parole
PROGRAMMI	Solo lettura						Rilocabile
LIMITI DI VELOCITA'	Tempo reale	Lento		Medio			Orientato al throughput
INGRESSO/ USCITA	Integrato	Pochi dispositivi semplici		Alcuni dispositivi complessi			Tutte le apparecchiatute
PROGETTO	Logico	Logica + microprogrammi		Micro e macro programmi			Macroprogrammi con linguaggi evoluti-sistemi di software
VOLUME DI PRODUZIONE	Grande						Piccolo

- Terminali
- Microcomputer

Ciascuna di queste categorie è discussa in dettaglio nel *Microprocessor Handbook*[3]. Come mostrato nella Tabella 1-1 e Fig. 1-1, le applicazioni dei microprocessori cadono tra la logica a relay e la logica random a componenti discreti (porte a flip-flop) da un lato e sistemi minicomputer a basso costo, come l'LSI-11 ed il PDP-8/A, dall'altro. I microcomputer costruiti a partire da microprocessore non sono certo così sofisticati come i più popolari minicomputer oggi sul mercato ed in genere non sono in grado di gestire facilmente situazioni in cui sia richiesta una elevata capacità di elaborazione dati. Attualmente i linguaggi di programmazione alto livello, tipo FORTRAN e COBOL, non sono disponibili per i microprocessori, non tanto per una bassa potenza quanto per il supporto software molto semplice, con cui sono corredati (solo ora sono disponibili i compilatori Basic e Cobol per microprocessori: N.D.T.).

Per ora sarebbe più appropriato chiamare i sistemi costruiti a partire dai microprocessori, *microcontrollori o processori logici,* data la facoltà con cui implementano sequenziatori programmabili e con cui elaborano dati logici, prendendo decisioni in funzione dei risultati dell'elaborazione stessa. Quindi, possono realizzare una data sequenza di operazioni in funzione delle caratteristiche dei dati in ingresso.

DOVE SI COLLOCANO I MICROCOMPUTER

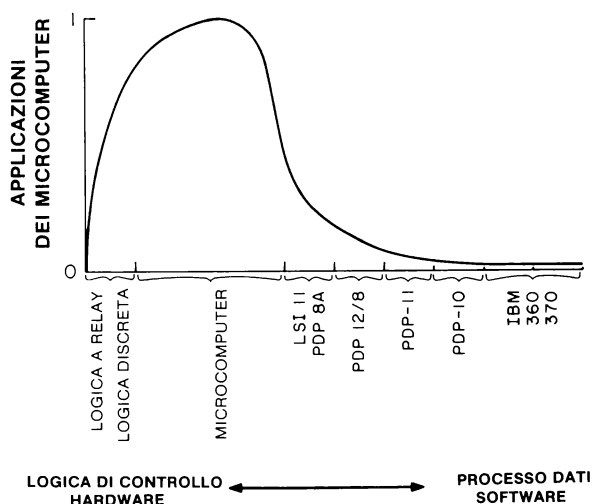



Fig. 1-1. Applicazioni previste per i microcomputer. I microcomputer si collocano tra la logica discreta ed i microcomputer a basso costo.

Quando fu introdotto per la prima volta, l'8080 era venduto singolarmente a 360 $. Alla fine del 1977, il costo è sceso a circa 10 $. Quindi d'ora in poi, il costo di un dispositivo basato sull'8080, ed in genere su un qualunque microprocessore, sarà funzione soprattutto delle parti meccaniche, ad esempio del contenitore e dei dispositivi elettromeccanici e non della parte «microprocessore».

In questo libro, si darà importanza alle applicazioni logiche e di controllo. Certo non è da dimenticare che molti microcomputer, con CPU 8080, corredati di periferiche da 5000 $, sono usati in competizione con sistemi quali il PDP-11. Piuttosto interesseranno situazioni in cui sistemi gerarchici mini-micro permettano di collegare ad un singolo minicomputer potente, più dispositivi basati sull'8080. Un esempio di questa gerarchia è indicata nella Fig. 1-2. Gli strumenti da A a G, comunicano tutti direttamente con un minicomputer. Tutti gli strumenti sono controllati da microprocessori 8080, il mini gestisce il dialogo ed acquisisce dati da tali sistemi periferici intelligenti. Il microcomputer può anche fornire i set point di controllo ai microcomputer all'inizio di ogni giornata.

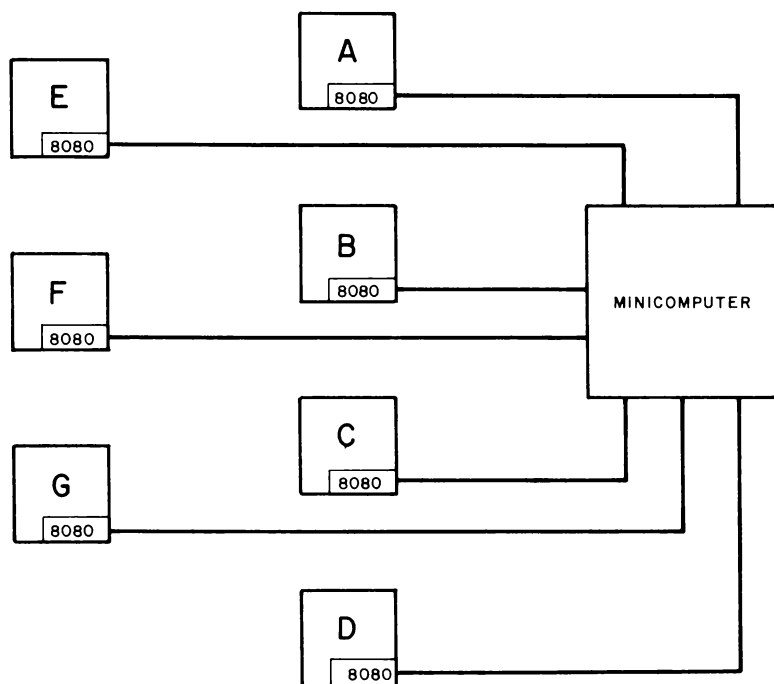



Fig. 1-2. Esempio di gerarchia di computer. I singoli strumenti o macchine, da A a G sono controllate da microprocessori 8080, posti al loro interno. Questi microprocessori comunicano nei due sensi con il minicomputer, che esegue il monitoraggio delle operazioni dell'intero sistema.

Ogni microcomputer può contenere da 2K ad 8K di memoria mentre il minicomputer può avere da 16 a 32K, per permettergli di lavorare con un linguaggio alto livello, quale il FORTRAN o l'APL. Ricordatevi che i futuri minicomputer saranno realmente a basso costo, forse non più di qualche centinaio di dollari.

Le teorie più avveniristiche parlano di costi sui 10/20 dollari per sistemi completi microcomputer in quantità di 100: a questo prezzo, è ovvio che saranno usati dappertutto.

DEFINIZIONI

Accumulator (Accumulatore) — Il registro e la associata circuiteria elettronica digitale nell'unità aritmetica/logica (ALU) di un computer, in cui vengono realizzate le operazioni logiche ed aritmetiche.

Bidirectional (Bidirezionale) — Operante in entrambe le direzioni.

Bidirectional date bus (Bus dati bidirezionale) — Un bus dati in cui le informazioni digitali possono essere trasferite in entrambe le direzioni.

Bus — Il bus è un canale sul quale sono trasferite le informazioni digitali, da una delle molte sorgenti verso una delle molte destinazioni. Può avvenire solo un trasferimento di informazioni alla volta; mentre tale trasferimento è in corso, tutte le altre sorgenti che sono legate al bus devono essere disabilitate.

Clock — (a) Qualsiasi dispositivo che genera almeno un impulso di clock, o (b) un dispositivo di timing in un sistema che fornisce una serie continua di impulsi di timing.

Computer — Qualunque dispositivo, in genere elettronico, in grado di accettare informazioni, di effettuare comparazioni, somme, sottrazioni moltiplicazioni, divisioni e successivamente in grado di fornire all'utente i risultati di queste operazioni in forma accettabile. Gli elementi principali sono la memoria, la sezione di controllo, la parte aritmetica e logica, l'Input/Output.[4]

Computer interfacing (Interfacciamento di un computer) — La sincronizzazione della trasmissione di dati digitali tra un computer ed uno o più dispositivi esterni di Input/Output.

Controller (Controllore)	Uno strumento che mantiene un processo ad una condizione a diversi livelli o stati, come determinato dal confronto del valore attuale con il valore desiderato.
Data processor	Dispositivo digitale che «processa» i dati, cioè li tratta, nel senso generico del termine. Può essere un computer, ed in genere acquisisce dati, li distribuisce, li analizza, ed attua operazioni di riorganizzazione dati. Queste operazioni non sono necessariamente di «calcolo». Data processor è un termine più inclusivo del termine computer.
Device select pulse (Impulso di selezione dispositivo)	Un impulso di sincronizzazione generato da un computer per sincronizzare le operazioni di uno specifico dispositivo di ingresso o uscita.
Digital computer (Computer digitale)	Un dispositivo elettronico che è capace di accettare, memorizzare e manipolare aritmeticamente le informazioni, che comprendono sia i dati che i programmi di controllo. L'informazione viene trattata sotto forma di cifre in codice binario (0 od 1) che sono rappresentate da due livelli di tensione.[6]
Digital controller (Controllore digitale)	Un controller che acquisisce il valore attuale di una condizione in forma digitale e lo paragona ad un valore desiderato contenuto nel controller. Se vi è una differenza tra i due, un segnale digitale è inviato all'esterno, per eliminare la differenza.
Direct address (Indirizzo diretto)	Un indirizzo che specifica la locazione di memoria di una istruzione o di un byte di dati.
External device addressing (Indirizzamento di un dispositivo esterno)	Un nome di un dispositivo, espresso come codice digitale, che è generato dalla CPU per indirizzare uno specifico dispositivo esterno. Si possono indirizzare dispositivi sia di ingresso che di uscita.
Fixed program computer (Computer a programma fisso)	Un computer in cui la sequenza delle istruzioni è permanentemente presente o «cablata». Il programma non è soggetto a cambiamento se non in base ad una specifica operazione di caricamento e cablaggio di una nuova sequenza di istruzioni.[5]
General-purpose computer (Computer general purpose)	Un computer progettato per risolvere una grande varietà di problemi, un computer che può essere adattato ad un'ampia classe di applicazioni.[5]

Hardware	L'insieme dei dispositivi meccanici, magnetici, elettrici ed elettronici, di cui è composto un computer; l'insieme del materiale che costituisce un computer.[2]
Input/Output	Termine generale indicante la strumentazione usata per comunicare con un computer, ed i dati coinvolti nella comunicazione.
Interfacing (Interfacciamento)	Il collegare elementi di un gruppo (come persone, dispositivi) in modo tale da metterli in grado di funzionare (agire) in modo compatibile e coordinato.
Interrupt (Interruzione)	In un computer, l'arresto della normale esecuzione di un programma in modo tale che il programma stesso possa essere ripreso al punto in cui era stato interrotto. La sorgente dell'interrupt può essere sia interna che esterna.
Memory (Memoria)	Qualsiasi dispositivo che può contenere 1 o 0 logico in modo tale da permettere l'accesso ad un bit o ad un gruppo di bit.[10]
Memory address (Indirizzo di memoria)	La locazione in cui è contenuta una parola di memoria.
Memory cell (Cella di memoria)	Un singolo elemento di memorizzazione.
Memory data (Dato di memoria)	La parola di memoria che occupa una specifica locazione, oppure le parole di memoria allocate, nel loro complesso, in una data memoria.
Memory word (Parola di memoria)	Un gruppo di bit che occupano una locazione in un dato computer. Questo gruppo è trattato dalla circuiteria del computer come una singola entità, dall'unità di controllo come un'istruzione e dall'unità aritmetica come una singola quantità. Ogni bit è posto in una singola cella di memoria.
Microcomputer	Un computer completamente operativo, basato su un chip microprocessore.
Microcontroller (Microcontrollore)	Un piccolo controller, piuttosto che non un controller basato su un microprocessore.
Microprocessor (Microprocessore)	Un singolo circuito integrato che contiene almeno il 75% della potenza di un piccolo computer.

Monostable multivibrator (Multivibratore monostabile) Un circuito che ha un solo stato stabile, da cui può essere forzato a cambiare, ma solo per un determinato intervallo di tempo dopo il quale ritorna allo stato originale.

Programmable read only memory (Memoria a sola lettura programmabile) Memoria a sola lettura, che è programmabile (field programmable) dall'utente.

Pulser (Generatore di impulsi) Uno switch che genera un impulso di clock.

Random access memory (Memoria ad accesso casuale) Una memoria a semiconduttore in cui possono essere scritte (stored) 0 logici ed 1 logici e successivamente letti.[10]

Read (Leggere) Trasmettere dati di una memoria a qualche altro dispositivo digitale.

Read only memory (Memoria a sola lettura) Una memoria a semiconduttore da cui dati digitali possono essere rapidamente letti ma in cui non è possibile scrivere, come nel caso della RAM.[10]

Software La totalità dei programmi e delle routine usate per estendere la capacità dei computer, come i compilatori, gli assemblatori, le routine, le subroutine. Tutto ciò di un computer che non è hardware.

Special purpose computer (Computer special purpose) Un computer progettato per risolvere una specifica classe od un limitato range di problemi.

Stored program computer (Computer a programma memorizzato) Un computer in grado di attuare sequenze di istruzioni internamente memorizzate, in genere con la possibilità di modificare queste istruzioni.[5]

Volatile memory (Memoria volatile) Tra i computer, qualsiasi memoria che mantiene le informazioni al permanere dell'alimentazione. E' l'opposto di memoria non volatile.[4]

Wired-program computer (Computer a programma cablato) Un computer in cui le istruzioni che indicano le operazioni sono specificate dalle disposizioni e dalle interconnesisoni dei collegamenti. I collegamenti sono realizzati da un pannello di controllo asportabile, il quale permette la flessibilità

delle operazioni: quanto detto si applica in realtà anche ai computer a programma fisso, ma la differenza è data dal fatto che mentre in un computer a programma cablato il programma può essere modificato da pannello, cioè le interconnessioni non sono fisse, nel caso del compatto a programma fisso i collegamenti non sono modificabili.[4]

Write (Scrivere) Trasmettere dati in memoria, da qualche dispositivo digitale. Sinonimo di Store.

REVISIONE DI ALCUNI CONCETTI DI ELETTRONICA DIGITALE

Due precedenti Bugbook, i *Bugbook I e II,* contengono i riferimenti necessari per un efficiente utilizzo di questo testo. Alcuni dei più importanti chip e concetti digitali di cui è necessario avere una conoscenza, sono:

- Le operazioni logiche AND, OR, NAND, OR Esclusivo.
- Le caratteristiche di gating delle quattro porte base: AND, NAND, OR, NOR.
- I circuiti integrati 7400, 7402, 7408, 7432.
- I decoder, specialmente il circuito integrato 7442 da 4 a 10 linee, ed il 74154 da 4 a 16 linee.
- I latch, comprendenti i circuiti integrati 7474, 7475, 74100, 74175, 74192, 74193, 74198. I circuiti integrati 74192 e 74193 sono contatori; il 74198 è uno shift register ad 8 bit.
- Flip-flop J-K, come i circuiti integrati 7470, 7473, 7476, 74106.
- Contatori, come i circuiti integrati 7490, 7493, 74192 e 74193.
- Sistemi di bus three-state.
- Dispositivi I/O (ingresso/uscita), come switch logici, generatori di impulsi, clock, indicatori a LED, display a LED 7 segmenti.
- I terminali *strobe, enable* (abilitare), *disable* (disabilitare).
- Multivibratori monostabili, come i circuiti integrati 74121, 74128, 74123 e 555.
- Logica sottoposta a clock.
- Registri, compreso il 74198 8 bit shift-register.
- Le operazioni aritmetiche di somma e sottrazione, e l'utilizzo del carry.
- I sistemi a multiplexer, come il 74153 da 4 ad 1 linea.
- Le funzioni degli ingressi di strobe e enable nei chip di circuiti integrati della serie 7400.
- I codici binario, BCD, ottale, esadecimale, ASCII.

ALCUNI UTILI RIFERIMENTI

Per quanto sia intenzione degli autori discutere dettagliatamente concetti di computer, minicomputer e microcomputer, questo capitolo è limitato. Per quei lettori interessati ad ampliare le loro nozioni, sono raccomandati i seguenti testi:

1 - *The Value of Power,* General Automation, Inc., 1055 South East Street, Anaheim, California 92805, 1973.
2 - Douglas Lewin, *Theory and Design of Digital Computers,* John Wiley & Sons, Inc., New York, 1972.
3 - David Hagelbarger and Saul Fingerman, *CARDIAC: A Cardboard Illustrative Aid to Computation,* Bell Telephone Laboratories, Inc., 1968.
4 - *The Microprocessor Handbook,* Texas Instruments, Inc., P.O. Box 5012, Dallas, Texas 75222.
5 - D. J. Woollons, *Introduction to Digital Computer Design,* Mc Graw-Hill, Inc., New York, 1972.
6 - Una serie di sei testi: (1) *Binary Arithmetic,* (2) *Microcomputer Architecture,* (3) *The 4-bit Microcomputer,* (4) *The 8-bit Microcomputer,* (5) *Assemblers and Prototyping Systems,* e (6) *8-bit Assemblers and Compilers,* Iasis, Inc., 770 Welch Road, Suite 154 ED, Palo Alto, California 94304.
7 - Jules Finkel, *Computer Aided Experimentation: Interfacing to Minicomputers,* John Wiley & Sons, Inc., New York, 1975.
8 - *Intel 8080 Microcomputer Systems User's Manual,* Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051, Luglio 1975.
9 - *An Introduction to Microcomputers, Volume 1: Basic Concepts,* Adam Osborne and Associates, Inc., P.O. Box 2036, Berkeley, California 94702.
10 - *8080 Programming for Logic Design,* Adam Osborne and Associates, Inc., P.O. Box 2036, Berkeley, California 94702.
11 - *Software Design for Microprocessors,* Texas Instruments, Inc., Dallas, Texas 75222.

DATA PROCESSOR, MICROPROCESSORE, MICROCOMPUTER

E' difficile trovare una buona definizione del termine *computer digitale.* Una buona analisi di cosa si intende per computer è data da Donald Eadie nel suo libro *Introduction to the Basic Computer:* «Questo capitolo serve come introduzione generale al campo dei dispositivi digitali: con particolare riferimento a quei dispositivi chiamati *computer,* o, più propriamente, *data processor.* Il termine data processor è più inclusivo in quanto le macchine moderne che cadono in questa definizione non solo calcolano nel senso usuale del termine, ma eseguono anche altre funzioni sui dati con cui interagiscono. Ad esempio un data processor può ottenere dati da varie sorgenti, porle in uscita, modificarli e, se è

il caso, stamparli. Nessuna di queste operazioni coinvolge le operazioni aritmetiche normalmente associate con un dispositivo di calcolo, ma il termine "computer" è ancora applicabile».

«Tuttavia, per i nostri scopi un computer è realmente un data processor. Anche le operazioni di data processing come il modificare i dati, possono richiedere semplici operazioni aritmetiche come la somma. Questo spiega perché è presente un certo grado di imprecisione nel nostro linguaggio e perché esiste di fatto una certa confusione tra i termini *computer* e *data processor*. I due termini sono usati così diffusamente, che spesso è necessario, a seconda dei casi, chiedere cosa esattamente significhino».[2]

Eadie definisce il termine *data processor* come segue:

Data processor — Dispositivo digitale che «elabora» i dati, cioè li tratta, nel senso generico del termine. Può essere un computer, ed in genere acquisisce dati, li distribuisce, li analizza ed attua operazioni di riorganizzazione dati. Queste operazioni non sono necessariamente di tipo «calcolo». Data Processor è un nome più inclusivo del termine computer. Si può tentare di definire il termine microprocessore come segue:

Microprocessor — Un data processor molto piccolo.

Iniziamo ad avere dei problemi quando tentiamo di eseguire una distinzione tra *microprocessore* e *microcomputer*. Così si legge nel *Microprocessor Handbook* della Texas Instruments:

«Questa lezione inizia con la parola "microprocessore". Per alcuni microprocessore significa microcomputer, per altri le parole microprocessore e microcomputer indicano cose differenti. Microprocessore è un termine generico che descrive un sistema elettronico molto piccolo, in grado di attuare specifici compiti. Il microcomputer è una applicazione dei microprocessori».[3]

Gli autori di questo libro sono d'accordo con quanto detto sull'handbook della Texas. Il microprocessore è un singolo circuito che contiene circa il 75% della potenza di un piccolo computer. In genere non può fare nulla senza l'aiuto dei chip di supporto e di memoria. Il microcomputer è un'entità autonoma basata sul chip microprocessore. Il microcomputer contiene memorie, Latch, contatori, dispositivi di I/O, buffer, alimentazione, oltre al microprocessore. Può essere un tipico «black box» con un singolo switch: OPERATE/RESET.

Altre indicazioni in proposito si possono ottenere da Laurence Altman, nel numero del 18 Aprile 1974 di *Electronics:*

«Cosa è un microprocessore... ma prima vediamo cosa non è. Un microprocessore non è un computer, ma una parte di esso. Per ottenere un computer partendo da un microprocessore occorre della memoria addizionale per il programma e circuiti di I/O, per poter dialogare con le periferiche.

«Poi ancora, microprocessore significa unità centrale di elabo-

razione microprogrammabile. Molti microprocessori sono controllati da microprogrammi, altri, no.

«Cos'è un microprocessore, quindi?: la parte di controllo e di processo di un piccolo computer o microcomputer. Ancora, il microprocessore rappresenta un processore non più realizzato con più circuiti LSI MOS, ma costituito da un singolo chip (single chip microprocessor) .Come tutti i processori di tutti i computer, i microprocessori possono realizzare operazioni logiche ed aritmetiche sui dati, in parallelo, sotto il controllo di un programma. La distinzione fondamentale dai processori dei minicomputer è data dall'uso di circuiti LSI con bassa potenza ed a basso costo; si distinguono anche dagli altri dispositivi LSI, dalla loro programmabilità».

«In breve, se un minicomputer ha la potenza di 1, il microprocessore, più la circuiteria di supporto, ha una potenza di 1/4. Al crescere della tecnologia LSI, diventeranno più potenti. Il futuro ci darà microprocessori bipolari single-chip e CMOS-on-sapphire». Ancora, si può riportare quanto scritto da Aldo Cavalcoli e Valerio Scibilla, nel numero del mese di Gennaio 1978 di Elettronica Oggi:

«Volendo dare una definizione non rigorosa ma efficace, il microprocessore può essere considerato come un calcolatore miniaturizzato: tutta la potenza di elaborazione di un calcolatore è stata posta in un unico circuito integrato. Questa miniaturizzazione è il risultato di una evoluzione della tecnologia nel campo della costruzione dei circuiti elettronici. Possiamo senz'altro affermare di essere attualmente alla quarta generazione di componenti: valvole, transistori, micrologici, microprocessori. La parola microprocessore ha una sua spiegazione: innanzi tutto con "micro" si indicano le ridotte dimensioni fisiche del componente; il termine "processore" è utilizzato per indicare quella specifica sezione di un sistema di elaborazione dati cui è demandato il compito di controllare l'esecuzione di tutte le operazioni di un calcolatore. Molto semplicemente, un calcolatore è suddiviso in tre sezioni base: il processore centrale, o unità centrale di controllo, la parte di memoria e la sezione di ingresso ed uscita dati, che collega il calcolatore con il mondo esterno. Si può senz'altro affermare che l'intelligenza di un calcolatore, intelligenza ovviamente guidata dal tecnico che fa eseguire operazioni al calcolatore tramite programmi, risiede nell'unità centrale. Proprio questa parte è stata miniaturizzata e resa disponibile come singolo circuito integrato. Questa disponibilità ha realizzato, e realizzerà sempre più, una rivoluzione nell'elettronica...».

HARDWARE E SOFTWARE

L'*hardware* ed il *software* sono termini importanti che saranno utilizzati spesso in questo capitolo. Occorre quindi definirli bene:

Hardware — L'insieme dei dispositivi meccanici, magnetici, elettronici ed elettrici, di cui è composto un computer; l'insieme del materiale che costituisce un computer.[2]

Software — La totalità dei programmi e delle routine usate per estendere la capacità dei computer, come i compilatori, gli assembler, le routine, le subroutine. Tutto ciò di un computer che non è hardware.[5]

In questo volume per prima cosa svilupperete una competenza hardware con riferimento all'interfacciamento di un microcomputer basato sul microprocessore 8080. Dopo questo primo passo, svilupperete una specifica competenza software che vi permetterà di utilizzare il vostro hardware con la più ampia gamma di strumenti e macchine. Al crescere della vostra esperienza sui microcomputer, imparerete che spesso scrivere un programma è un'operazione abbastanza lunga, con riferimento, ad esempio, a una routine di alcune centinaia di passi.

COMPUTER E COMPUTER DIGITALI

E' a questo punto necessario definire i termini *computer* e *computer digitale.*

Sono comunque difficili delle buone definizioni, come già detto.

Computer: qualunque dispositivo, in genere elettronico, in grado di accettare informazioni, di effettuare comparazioni, somme, sottrazioni, moltiplicazioni, divisioni e successivamente in grado di fornire all'utente i risultati di queste operazioni, in forma accettabile. Gli elementi principali sono la memoria, la sezione di controllo, la parte aritmetica e logica, l'Input/Output.[4]

Un dispositivo in grado di accettare informazioni, di attuare determinate elaborazioni delle informazioni ed infine di fornire il risultato delle operazioni; è in genere formato da dispositivi di input, di output, di memorizzazione, di unità aritmetico-logiche e di controllo.[5]

Computer digitale: un dispositivo elettronico che è capace di accettare, memorizzare e manipolare aritmeticamente le informazioni, che comprendono sia i dati che i programmi di controllo. L'informazione viene trattata sotto forma di cifre in codice binario (0 e 1) che sono rappresentate da due livelli di tensione.[6]

Altra definizione è: un computer che elabora informazioni rappresentate da una combinazione di dati discreti e discontinui, in confronto ai computer analogici che lavorano su dati continui. Un dispositivo per attuare sequenze di operazioni aritmetiche e logiche in base ad un programma, (sequenza di istruzioni in esso memorizzate) in opposizione a quei «calculator» (card-programmed calculator) in cui la sequenza è introdotta di volta in volta manualmente.[5]

Dopo queste definizioni, ne occorrono altre, e cioè:

Computer a programma fisso: un computer in cui la sequenza dell'istruzione è permanentemente presente o «cablata». Il programma non è soggetto a cambiamenti se non in base ad una specifica operazione di caricamento e cablaggio di una nuova sequenza di istruzioni.[5]

Computer general purpose: un computer progettato per risolvere una grande varietà di problemi; un computer che può essere adattato ad un'ampia classe di operazioni.[5]

Computer special purpose: un computer progettato per risolvere uno specifico problema.[5]

Computer a programma memorizzato: un computer in grado di attuare sequenze di istruzioni internamente memorizzate, in genere con la possibilità di modificare queste istruzioni.[5]

Computer a programma cablato: un computer in cui le istruzioni che indicano le operazioni sono specificate dalla disposizione e dalle interconnessioni dei collegamenti. I collegamenti sono realizzati da un pannello di controllo asportabile, il quale permette la flessibilità delle operazioni; quanto detto in realtà si applica anche ai computer a programma fisso, ma la differenza è data dal fatto che mentre in un computer a programma cablato, il programma può essere modificato da pannello, cioè le interconnessioni non sono fisse, nel caso del computer a programma fisso, i collegamenti sono non modificabili.[4]

COSA E' UN CONTROLLORE

Graf ha definito il *controllore* come segue:

Controllore: uno strumento che controlla un processo e lo condiziona ad un livello desiderato in base a confronti tra il valore attuale ed il valore voluto.[4]

I controllori possono essere analogici o digitali ed ancora possono essere elettronici, meccanici ed anche pneumatici o combinazioni di essi.

Un *controllore digitale* acquisisce il valore attuale della condizione in forma digitale ed effettua un confronto con il valore voluto contenuto entro il controllore. Se sussiste una qualche differenza fra i due, un segnale digitale è inviato all'esterno del dispositivo per iniziare le opportune azioni tese a ridurre tale differenza. Il controllore digitale è costituito da circuiti integrati e componenti discreti montati su piastre di circuito stampato, oppure è un computer di qualsiasi dimensione con un numero limitato di chip che servono da interfaccia tra esso ed il mondo esterno.

La questione del costo diventa un fatto importante quando si considera l'uso dei computer e dei controllori. Non è certo pos-

sibile controllare 100 dispositivi, del valore di 500.000 lire ciascuno, con un computer da un miliardo: sarebbe una situazione di macroscopico sovradimensionamento. D'altra parte un grosso computer può senz'altro controllare un impianto chimico del valore di 20 miliardi. Quindi si può giustificare il costo di un computer solo se esso rappresenta una modesta percentuale del costo globale dell'impianto e del sistema da controllare. Tenete presente che con l'avvento dei microcomputer, i costi della parte di controllo dovrebbero diminuire in modo sensibile.

UN TIPICO MICROCOMPUTER BASATO SULL'8080

Un tipico microcomputer ottenuto a partire dal chip 8080, è mostrato nella Fig. 1-3. Questo microcomputer è in grado di effettuare tutte le operazioni tipiche di un computer. Ad esempio:

- Può realizzare l'Input e l'Output dei dati.
- Contiene un'unità aritmetica-logica (ALU), posta entro il chip 8080, in grado di attuare le principali operazioni logiche ed aritmetiche.
- Contiene una memoria «veloce» (gli autori ritengono che la velocità è un importante requisito per un computer).
- E' programmabile, secondo sequenze di dati ed istruzioni del tutto libere.
- E' digitale.

La Fig. 1-3 mostra il flusso dati del microcomputer. Nella sezione seguente discuteremo il diagramma e le parti individuali del flusso dati.

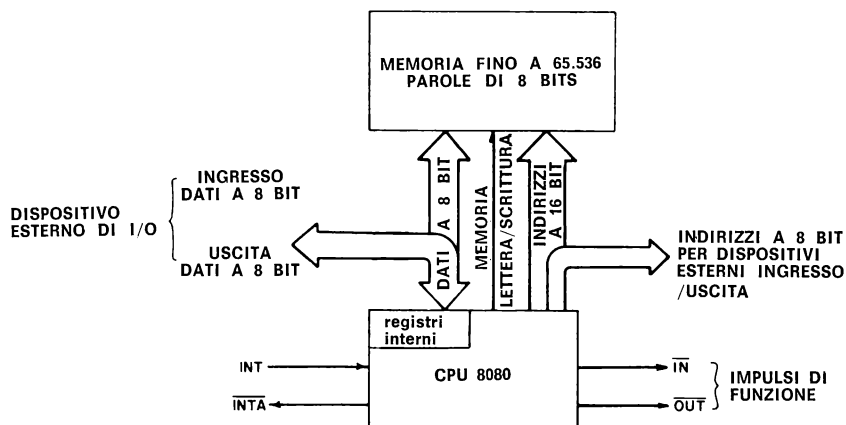



Fig. 1-3. Un tipico microcomputer 8080 in cui sono indicati i flussi dei dati più significativi.

Memoria

Consideriamo all'inizio la comunicazione dati tra l'unità centrale 8080 (CPU) e la memoria. Vi servono alcune definizioni che saranno utili nel proseguimento:

Memory (memoria): qualsiasi dispositivo che può contenere 1 o 0 logico in modo tale da permettere l'accesso ad un bit od a un gruppo di bit.[10]
Memory address (indirizzo di memoria): la locazione in cui è contenuta una parola di memoria.
Memory cell (cella di memoria): un singolo elemento di memorizzazione.
Memory data (dato di memoria): la parola di memoria che occupa una specifica locazione, oppure le parole di memoria allocate, nel loro complesso, in una data memoria.
Memory word (parola di memoria): un gruppo di bit che occupano una locazione in un dato computer. Questo gruppo è trattato dalla circuiteria del computer come una singola entità, dall'unità di controllo come un'istruzione e dall'unità aritmetica come una singola quantità. Ogni bit è posto in una singola cella di memoria.
Programmable read only memory (PROM) (memoria a sola lettura programmabile): memoria a sola lettura, che è programmabile (field programmable) dall'utente.[10]
Read/write memory (memoria a lettura e scrittura): una memoria a semiconduttore in cui possono essere scritte (stored) 0 logici ed 1 logici e successivamente letti.[10] Viene anche chiamata RAM.
Read only memory (ROM) (memoria a sola lettura): una memoria a semiconduttore da cui dati digitali possono essere rapidamente letti, ma in cui non è possibile scrivere, come nel caso della RAM.[10]
Read (leggere): trasmettere dati dalla memoria a qualche altro dispositivo elettronico.
Volatile memory (memoria volatile): nei computer, qualsiasi memoria che è in grado di mantenere le informazioni solo finché l'alimentazione è applicata alla memoria. L'opposto di non volatile memory (memoria non volatile).[4]
Write (scrivere): trasmettere dati in memoria da qualche dispositivo digitale. E' sinonimo di *store.*

Il microprocessore 8080 usa parole di 8 bit; queste parole sono collocate in memoria e sono indirizzate tramite un address bus a 16 bit. Con un rapido calcolo si può verificare che il microprocessore può accedere a $2^{16} = 65.536$ differenti locazioni di memoria. Questo accesso alla memoria è diretto, cioè non dovete coinvolgere particolari procedure, né vi serve una particolare circuiteria addizionale, per avere accesso ad una delle 65.536 loca-

zioni. I circuiti integrati a 40 pin hanno i loro vantaggi e l'avere un pin per ognuna delle 16 linee di indirizzamento, è uno di questi vantaggi. La capacità totale di memoria del microprocessore 8080 viene indicata in termini di 64K. Questa capacità è ben superiore a quella necessaria per qualsiasi applicazione, ma è interessante sapere di avere questa potenza di riserva.

Una data locazione di memoria è indirizzata tramite l'*address bus a 16 bit*, come indicato nella Fig. 1-3. E' necessario solo 1 μs per realizzare tale O. L'address bus è mostrato nella Fig. 1-4.

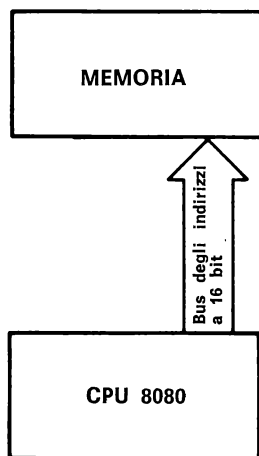



Fig. 1-4. Il bus degli indirizzi di memoria a 16 bit tra la CPU 8080 e la memoria.

I dati sono trasferiti tra l'8080 e la memoria tramite due bus ad 8 bit, il *data input bus* ed il *data output bus*, entrambi indicati nelle Figg. 1-3 ed 1-5. Nei più recenti microcomputer basati sul-

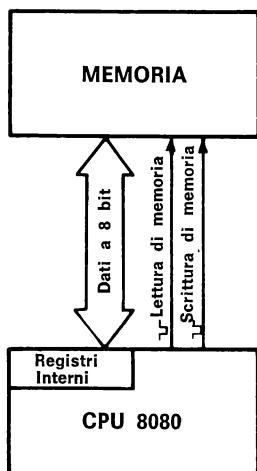



Fig. 1-5. Trasferimento dati tra la CPU 8080 e la memoria. In molti sistemi 8080 vi è un singolo bus dati bidirezionale.

l'8080, questi due bus sono combinati in un singolo *data bus bidirezionale* ad 8 bit. Col termine *input (ingresso)*, intendiamo ingresso nella CPU. Con *output (uscita)*, intendiamo uscita dalla CPU. Il punto di riferimento è sempre la CPU.

Il dato che lascia la CPU è sempre considerato come *data output (dato di uscita)*; il dato che entra nella CPU è sempre considerato come *data input (dato in ingresso)*. In alcuni casi, i dati in ingresso ed uscita sono trasferiti tra l'*accumulatore* e la memoria. Il termine *accumulatore* è definito nel seguente modo:

Accumulatore: il registro e la circuiteria elettronica associate nell'unità aritmetica di un computer, in cui vengono effettuate operazioni aritmetiche e logiche.

I dati possono essere trasferiti anche verso altri *registri* interni al microprocessore 8080. Un registro è definito nel seguente modo:

Registro: una circuiteria elettronica di memorizzazione, avente la capacità di una sola parola[4].

Altri registri, oltre all'accumulatore, sono ad esempio:

Instruction register (registro istruzione), che realizza la decodifica delle istruzioni.

6 registri general purpose, indicati dalle lettere B, C, D, E, H, L.

Il *registro program counter (contatore di programma)*.

Il *registro stack pointer.*

Almeno *3 registri temporanei*, ai quali non è possibile accedere. Tutti questi registri presentati con lo scopo di chiarirvi che i dati dalla memoria non sono trasferiti solo verso l'accumulatore. L'accumulatore comunque resta il cuore del microcomputer; le operazioni aritmetiche e logiche sono sempre realizzate coinvolgendo i dati presenti in accumulatore.

Non è possibile sommare direttamente il contenuto di una locazione di memoria a quella di un'altra locazione direttamente. Occorre sempre coinvolgere, in fase intermedia, l'accumulatore.

L'accumulatore è importante anche perché tutti i dati in ingresso ed uscita passano attraverso di esso, ogni qualvolta utilizzate le istruzioni di IN ed OUT. Tra la CPU 8080 e la memoria, esiste una singola linea di uscita, mostrata nella Fig. 1-3, detta *memory read/write (lettura/scrittura in memoria)*. Quando questa linea è allo stato logico 1, voi siete in grado di leggere dati nella CPU sia dalla memoria che da dispositivi esterni. Quando questa linea è allo stato logico 0, siete in grado di scrivere dati dalla CPU in memoria o in un qualsiasi dispositivo esterno. In alcuni sistemi si usano separate linee di lettura e scrittura. Infine, potete realizzare qualsiasi tipo di dispositivo elettronico digitale di memoria veloce, inclusa la memoria lettura/scrittura (R/W), a sola lettura (ROM), a sola lettura programmabile (PROM).

Memoria «veloce» significa semplicemente che la memoria può

realizzare operazioni sia di lettura che di scrittura durante una singola istruzione del microprocessore. Un tipico 8080 opera con un clock di 2 MHz, ed un'operazione di lettura scrittura necessaria solo di 650 ns. Quindi le memorie R/W, ROM, PROM, tutte necessitano di un tempo di accesso di 650 ns, per permettere l'attuazione completa dei vantaggi connessi con la massima velocità del clock. Possono essere usate memorie più lente, ma il microcomputer dovrebbe, in tal caso, attendere il verificarsi delle operazioni di lettura e scrittura.

Uscita dei Dati

Il *bus* ad 8 bit di *output data,* tra l'8080 e la memoria, serve anche come bus di uscita dei dati verso un dispositivo esterno. Questo è indicato nelle Figg. 1-3 ed 1-6.

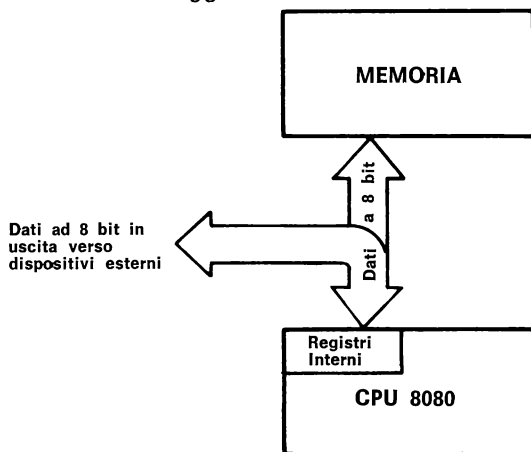



Fig. 1-6. Trasferimento dati tra la CPU 8080 e un dispositivo di uscita.

Quando si inviano dati verso un dispositivo esterno, vi sono alcune importanti condizioni da rispettare:

- Dovete selezionare lo specifico dispositivo di uscita che riceverà gli 8 bit del dato dall'accumulatore che è nella CPU 8080.
- Dovete indicare a questo dispositivo il preciso istante in cui il dato sarà disponibile sul *bus di uscita.*
- Il dispositivo deve «catturare» o «sottoporre a latch» questo dato, in un breve periodo di tempo, ad esempio 500 ns.

Le 3 specifiche indicate sono soddisfatte contemporaneamente con un singolo impulso di uscita, generato dall'8080, con il supporto di alcuni chip esterni all'8080 stesso.

Questo impulso è chiamato *device select pulse (impulso di selezione dispositivo)*. Esso sincronizza l'8080 con i dispositivi esterni in modo tale che quando il microprocessore è pronto per fornire dati in uscita, il dispositivo esterno è pronto a riceverli. Ricordatevi bene che il microprocessore lavora a 2 MHz. Ogni istruzione è eseguita in un tempo molto rapido, da 2 a 9 μs. Di conseguenza, i dati contenuti nell'accumulatore, dati in uscita, non sono disponibili per molto tempo.

Voi dovete catturare questi dati in 500 ns, altrimenti li perdete. Non possiamo enfatizzare quanto sarebbe necessario, quanto è importante questa rapida «cattura» dei dati in uscita, ai fini del successo delle operazioni di ingresso-uscita dati. Tratteremo questi argomenti in modo molto dettagliato, quando tratteremo l'interfacciamento. Se siete interessati a capire come l'impulso $\overline{OUT}$, utilizzato per generare i device select pulse, sia creato dal microprocessore 8080, andate direttamente alla sezione intitolata «Identificazione dei cicli macchina» nel Capitolo 6.

Ingresso dei Dati

Il *bus* di 8 bit di *input data,* fra la CPU 8080, e la memoria, viene utilizzato anche come bus di ingresso dei dati che provengono da eventuali dispositivi esterni. Quanto detto è mostrato nelle Figg. 1-3 ed 1-7. Le considerazioni base applicate all'output data, valgono anche per l'input data, cioè:

- Dovete selezionare lo specifico dispositivo di ingresso, che trasmetterà i dati all'accumulatore entro la CPU.
- Dovete indicare a questo dispositivo il primo istante in cui il bus di ingresso è pronto ad acquisire i dati e trasferirli all'accumulatore.

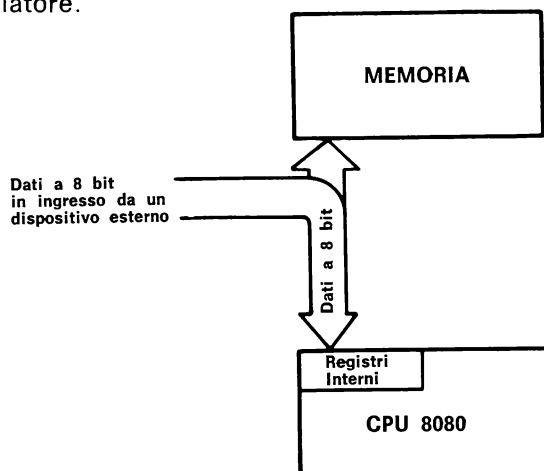



Fig. 1-7. Trasferimento dati tra un dispositivo di ingresso e la CPU 8080.

- L'accumulatore deve «catturare» i dati in un tempo molto breve, tipicamente 500 n/sec.

Le tre funzioni indicate sono realizzate nello stesso tempo tramite un singolo impulso in uscita, generato dall'8080 con il supporto di alcuni chip esterni. Tale impulso è anche chiamato *device select pulse (impulso di selezione dispositivo)*. Esso sincronizza la CPU 8080 ed il dispositivo d'ingresso in modo che quando la CPU è pronta a ricevere i dati, anche il dispositivo è pronto a trasmetterli.

Altre Tecniche di Ingresso/Uscita

Vi è una tecnica alternativa e molto interessante per trasmettere i dati *in modo bidirezionale tra i registri B, C, D, E, H e L, e un dispositivo di ingresso/uscita.* Questa tecnica è chiamata *memory mapped I/O* ed i chip utilizzati fanno parte di una famiglia di chip di interfaccia recentemente commercializzata da molte case. Di questa famiglia fa parte l'8255 programmable peripheral interface, che trasforma un dispositivo di ingresso-uscita in una pseudo-locazione di memoria, indirizzata non con le istruzioni IN ed OUT, ma con istruzioni con riferimento in memoria, come la MOV, STA, LDA ed altre simili. Il vantaggio che si ottiene consiste nel salvare parecchi microsecondi per il trasferimento dati. In più, si semplifica il software. La tecnica è particolarmente adatta ai casi in cui si debba effettuare un'acquisizione od un trasferimento di blocchi di dati in brevi intervalli di tempo. Non discuteremo questa tecnica nel presente volume.

Indirizzamento di Dispositivi Esterni

L'*indirizzamento di dispositivi esterni* può essere definito come l'utilizzo del software per generare impulsi di I/O di sincronizzazione, detti *device select pulses,* atti a sincronizzare il trasferimento dati tra la CPU ed il mondo esterno. Questa è una delle funzioni più importanti dell'interfacciamento, e deve perciò essere studiata attentamente.

L'obiettivo dell'indirizzamento di dispositivi esterni è quello di *generatore un singolo impulso di clock in un preciso istante verso uno specifico dispositivo esterno di ingresso ed uscita.*

L'impulso di clock può essere sia positivo che negativo. Con il microprocessore 8080, è più facile generare impulsi negativi. Questo è realizzato decodificando un *codice dispositivo* ad 8 bit, che è presente per 1,5 µs nell'address bus, ed utilizzando una funzione di I/O od impulso di sincronizzazione, detta $\overline{IN}$ e $\overline{OUT}$, che è presente per 500 ns in corrispondenza dell'intervallo di 1,5 µs associato con il codice dispositivo. I due impulsi e l'indirizzo ad 8 bit per l'indirizzamento di dispositivi esterni di I/O, è mostrato nelle Figg. 1-3 ed 1-8. Vedremo nel Capitolo 4 i detta-

gli relativi alla generazione dei device select pulses; diamo qui un breve riassunto.

L'istruzione per il trasferimento dei dati tra l'accumulatore ed il dispositivo esterno di I/O, specifica il dispositivo desiderato. Per un'istruzione di output, potete scegliere tra 256 differenti codici di dispositivi. Lo stesso vale per le istruzioni di input. Quindi è possibile inviare impulsi di selezione a 256 differenti dispositivi di ingresso ed a 256 differenti dispositivi di uscita. Come si realizza ciò? Ogni istruzione di I/O contiene un *device code (codice dispositivo)* ad 8 bit. Quindi: $2^8 = 256$ differenti dispositivi. Utilizziamo una coppia di decoder da 4 a 16, come i decoder 74154, per decodificare il device code ad 8 bit in un singolo impulso. Diciassette 74154 possono generare 256 differenti impulsi, come sarà indicato nel Capitolo 4. I rimanenti dettagli saranno discussi utilizzando la Fig. 1-9. Questa fornisce i diagrammi di tempo per i contenuti dell'accumulatore, l'address bus, e per l'impulso $\overline{OUT}$ generato dall'istruzione di OUT, con l'ausilio di qualche chip di supporto. Considerate attentamente quanto segue:

- I contenuti dell'accumulatore sono disponibili per alcuni microsecondi, e forse di più.
- Il codice dispositivo ad 8 bit, si presenta sugli 8 bit meno significativi dell'address bus a 16 bit, per un periodo di 1,3 µs. Nella Fig., il codice è 11010001_2, cioè 321_8 in ottale.
- L'impulso di sincronizzazione $\overline{OUT}$ dura solo 500 ns. *E' proprio durante questi 500 ns che i dati sono trasferiti tra l'accumulatore ed il dispositivo in uscita.*

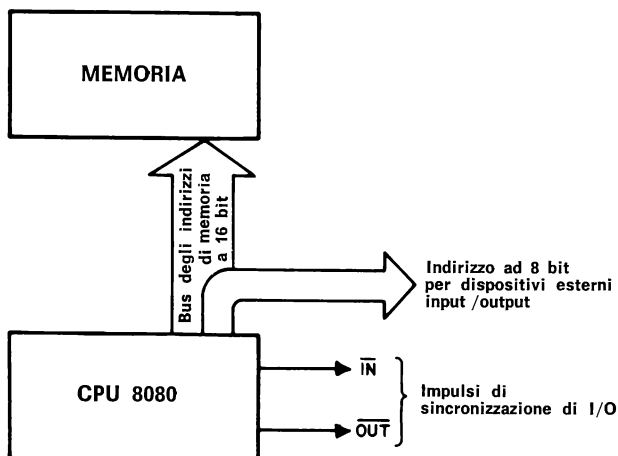



Fig. 1-8. Durante una istruzione di input o di output per l'8080, sul bus degli indirizzi appare un codice dispositivo ad 8 bit, e sul bus di controllo appare un impulso di sincronizzazione di I/O. Tali segnali sono usati per abilitare il trasferimento dei dati di I/O.

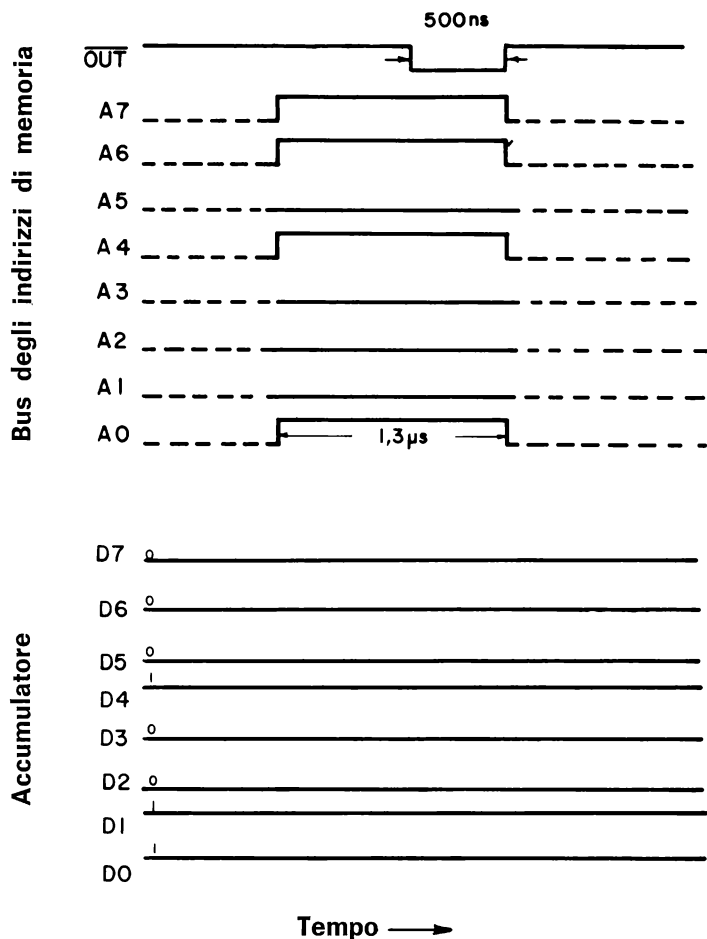



Fig. 1-9. Diagramma di tempo che indica gli stati logici presenti sul bus degli indirizzi, entro l'accumulatore e sulla linea $\overline{OUT}$, durante una istruzione di OUT.

Sia il codice ad 8 bit che l'impulso di sincronizzazione $\overline{OUT}$, sono connessi ai decoder 74154, al fine di generare un singolo codice dispositivo per il dispositivo 321_8, per un periodo di 500 ns. Quindi, l'uso del termine «catturare», è veramente appropriato, per un trasferimento dati tra l'accumulatore ed il mondo esterno.

L'ingresso di 8 bit di dati nell'accumulatore tramite un'istruzione IN, procede lungo direttive simili a quelle indicate nella Fig. 1-5. La sola differenza è che voi dovete sostituire il termine buffer data per l'accumulatore, ed $\overline{IN}$ per $\overline{OUT}$ nella figura. Il trasferimento dati verso l'accumulatore si realizza in soli 500 ns.

Interrupt Servicing

L'ultima cosa che vogliamo discutere è associata con la tecnica relativa al *servizio dell'interrupt*. Il termine *interrupt* può essere definito nel seguente modo:

Interrupt — In un computer, l'arresto della normale esecuzione di un programma, in modo tale da permettere al programma stesso di essere ripreso nel punto esatto in cui era stato interrotto. La sorgente dell'interrupt può essere interna ed esterna.

La questione importante è: perché vogliamo interrompere un programma in corso? La risposta è che questo è il metodo più efficiente per operare su un microcomputer o, con riferimento a questo argomento, su qualunque computer. *Fino a quando un dato dispositivo esterno di ingresso/uscita non richiede assistenza, o servizio, da un microcomputer, è più efficiente per il microcomputer ignorare completamente il dispositivo.* Infatti il microcomputer può essere programmato in modo da ignorare tutti i dispositivi esterni; resta in un «wait loop» (ciclo di attesa), mentre aspetta un segnale di interrupt da uno dei dispositivi esterni. Se più dispositivi richiedono servizio nello stesso tempo, il microcomputer possiede un protocollo sia hardware che software per individuare quale dei possibili dispositivi è più importante. Assegnerà una priorità ad ogni dispositivo e sempre servirà il dispositivo a più alta priorità, per primo.

Quando è generato un interrupt, si realizza la seguente sequenza:

1 - Il computer memorizza l'indirizzo di memoria dell'istruzione seguente a quella che è in esecuzione in quell'istante.

2 - Il computer memorizza tutte le informazioni temporanee — flag, contenuto dell'accumulatore e degli altri registri — che può essere importante avere quando il programma interrotto riassume il controllo.

3 - Il computer va verso una ben precisa locazione di memoria ed eseguire una serie di passi di programma per «servire» il dispositivo interrompente.

4 - Una volta esaurito il «servizio», il computer richiama le informazioni temporanee e ritorna alla sezione di programma seguente a quello in cui si era verificata l'interruzione.

Questo è simile alla situazione che si verifica quando venite interrotti mentre state leggendo un libro, ad esempio questo. Dovete allora segnare dove siete arrivati, ricordare qualunque speciale informazione, e poi dare la vostra attenzione all'«interruzione». Dopo il servizio, dovete ripristinare le informazioni da voi memorizzate, localizzare il segno e continuare la lettura.

Se più di un dispositivo di ingresso/uscita richiede l'interrupt,

il microcomputer deve decidere quale servire per primo. Presa la decisione, ricorda quali restanti dispositivi hanno generato le richieste di interrupt e poi procede alla subroutine di servizio.

Nel microcomputer 8080, un'istruzione ad 8 bit è forzata nella CPU all'istante in cui si verifica l'interrupt, per indicare al computer deve andare in memoria, per servire l'interrupt. Sono a disposizione solo 8 differenti indirizzi di memoria, certo non molti.

Vi sono svariati trucchi per gestire gli interrupt. Lo schema di Fig. 1-10 indica i segnali importanti durante una richiesta di interrupt; un impulso di interrupt, un segnale di controllo di riconoscimento interrupt, $\overline{\text{INTA}}$, ed una istruzione ad 8 bit forzata nel registro istruzione.

TEST

Questo test verifica la vostra comprensione del microcomputer e dei concetti di elettronica digitale descritti in questo capitolo. Scrivete per favore la vostra risposta su un foglio a parte.

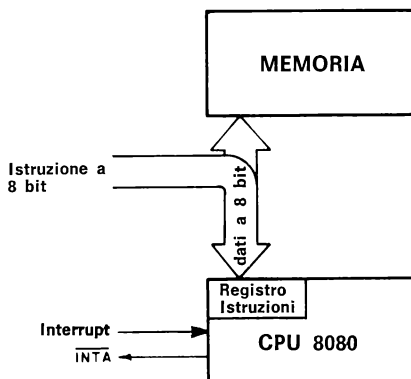



Fig. 1-10. Ricevendo una richiesta di interrupt, la CPU 8080 genera un segnale di controllo di «interrupt acknowledge», $\overline{\text{INTA}}$, usato per abilitare una istruzione da 8 bit nel registro istruzioni entro la CPU.

1-1 Spiegate la differenza tra un microprocessore e microcomputer.

1-2 Tracciate un diagramma e mostrate i flussi dati più importanti in un tipico microcomputer basato sull'8080.

1-3 Quali sono i requisiti minimi per un computer del 1977?

1-4 Indicate e descrivete i quattro punti fondamentali dell'interfacciamento.

1-5 Con vostre parole, definite i seguenti termini:

Data processor
Controllore

Bus
Accumulatore
Flag
Indirizzo di memoria
Cella di memoria
Parola di memoria
Hardware
Software
Lettura
Scrittura
Interfacciamento
Interrupt
Impulso selezione dispositivo
Clock
Bus dati bidirezionali
Memoria lettura/scrittura
Memoria a sola lettura (ROM)

La vostra risposta sarà accettabile se risulterà corretta e se impiegherete 90 minuti a libro chiuso.

COSA AVETE REALIZZATO IN QUESTO CAPITOLO?

Si era detto, all'inizio del capitolo, che alla fine sareste stati in grado di:

- Spiegare le differenze tra un microprocessore ed un microcomputer.
  Un microprocessore è un circuito integrato single chip, mentre un microcomputer è un computer completo dal punto di vista operativo. Questa distinzione è stata discussa facendo riferimento ad esempi provenienti da varia letteratura.
- Definire i termini: computer, computer digitale, data processor, controllore, hardware, software, memoria, parola di memoria, indirizzo di memoria, dato di memoria, lettura, scrittura, memoria lettura/scrittura, memoria a sola lettura, interfacciamento, impulso selezione dispositivo, interrupt.
  Le definizioni per questi termini sono state date nel capitolo.
- Descrivere l'operatività di un tipico microcomputer 8080.
  Questo è stato fatto non molto dettagliatamente, comunque dovreste essere in grado di farlo abbastanza bene.

CAPITOLO 2

Un Piccolo Microcomputer Basato sull'8080

In questo capitolo vedrete come un microprocessore 8080 (o 8080A) può essere usato per realizzare un piccolo microcomputer. Esamineremo i segnali che entrano ed escono dal chip, vedremo i componenti di supporto, quali l'8224, usati per controllare il funzionamento del microcomputer, lo sviluppo dei bus degli indirizzi, dei dati e di controllo, che rappresentano parti vitali nelle applicazioni di interfacciamento.

OBIETTIVI

Alla fine di questo capitolo, sarete in grado di:

- Identificare il bus degli indirizzi di memoria, il bus dati, gli ingressi e le uscite di controllo, le alimentazioni, sul microprocessore 8080A.
- Descrivere le funzioni di ogni pin del microprocessore 8080A.
- Descrivere in dettaglio le varie sezioni che costituiscono un piccolo sistema a microcomputer.
- Elencare i principi generali relativi all'interfacciamento di un computer.
- Spiegare cosa è un dispositivo di I/O.
- Elencare tre usi importanti degli impulsi di selezione dispositivo.
- Elencare gli ingressi dei componenti della serie 7400, che possono essere sottoposti a strobe tramite gli impulsi di selezione dispositivo generati da un microcomputer.

DEFINIZIONI

Bit — Abbreviazione di «binary digit» (digit binario). Una unità di informazione equivalente ad una decisione binaria.

Bootstrap — Una tecnica od un dispositivo progettato per portarsi da solo in un desiderato stato tramite sue proprie azioni. Ad esempio una routine le cui prime poche istruzioni sono sufficienti per portare il resto della stessa routine nel computer, da un dispositivo di ingresso.[4]

Bus driver — Generalmente si riferisce ad un circuito integrato speciale, che è aggiunto al sistema di data bus per facilitare le operazioni di pilotaggio della CPU, quando più memorie sono collegate alla linea di data bus. Qualsiasi dispositivo a semiconduttore che incrementa le caratteristiche di assorbimento di corrente di ciascuna linea in un bus.[14]

Byte — Una sequenza di digit binari, che può essere eguale oppure più corta di una parola, trattata come una unità. Per l'8080A un byte è un gruppo di otto bit contigui che occupano una sola locazione di memoria.

Computer interfacing (Interfacciamento di un computer) — La sincronizzazione di una trasmissione dati digitali tra un computer ed uno o più dispositivi esterni di I/O.

Flag — In un computer, l'indicazione che una particolare operazione è stata completata.[4] Un flag tipicamente è un flip-flop che può essere settato o azzerato (resettato, cleared) in risposta ad operazioni che si verificano nel microcomputer.

HI memory address (Indirizzo di memoria HI) — Gli otto bit più significativi nella parola d'indirizzo di memoria a 16 bit, per il microprocessore 8080. Abbreviato con H o HI.

Interfacing (Interfacciamento) — Il collegamento di membri di un gruppo (come persone, strumenti, ecc.) in modo tale da renderlo in grado di lavorare, funzionare in modo compatibile e coordinato.

I/O (Ingresso/Uscita) — Abbreviazione di Input/Output[4] (ingresso/uscita).

I/O device (Dispositivo di ingresso/uscita)	Un lettore di nastro magnetico, stampante o dispositivo similare, che trasmette o riceve dati da un sistema di elaborazione dati o unità di memorizzazione secondaria.[4] Qualsiasi dispositivo digitale che trasmette o riceve dati da un computer.
Latch	Un semplice elemento di memorizzazione logica, come un flip-flop, usato per mantenere uno stato logico.
LO address memory (Indirizzo di memoria LO)	Gli altri bit meno significativi nella parola d'indirizzo di memoria a 16 bit, per il microprocessore 8080. Abbreviato con L o LO.
Memory address (Indirizzo di memoria)	Per il microprocessore 8080A, il numero a 16 bit che specifica la precisa locazione di memoria di una parola di memoria fra le 65536 possibili.
Status bit (Bit di stato)	Un singolo bit di informazione in uscita, posto sul bus dati durante l'esecuzione di un ciclo macchina e sottoposto a latch da un chip detto status latch. Una volta acquisito dal latch, può essere usato per controllare eventi esterni relativi al successivo ciclo macchina.
Status byte (Byte di stato)	Un byte che contiene 8 differenti bit di stato.
Status latch (Latch di stato)	Un circuito integrato, come ad esempio il 74174, latch a 6 bit, che attua il latch dei bit di stato, al loro apparire sul bus dati.
Sync	Abbreviazione per sincrono, sincronizzazione, ecc.[4]
Synchronize (Sincronizzare)	Porre un elemento di un sistema, al passo con un altro.[4]
Synchronization pulses (Impulsi di sincronizzazione)	Impulsi originati dalla strumentazione di trasmissione ed inviati alla strumentazione di ricezione, allo scopo di mantenerle entrambe al passo.
Synchronous (Sincrono)	Al passo o in fase; valido per dispositivi o macchine. Termine applicabile anche ad un computer, nel quale quanto realizzato da una sequenza di operazioni è controllato da un segnale di clock o impulsi.[4]
Synchronous computer (Computer sincrono)	Un computer digitale in cui tutte le operazioni ordinarie sono controllate da segnali pervenuti da un clock master.[4]

Synchronous inputs (Ingressi sincroni)	Quegli ingressi di un flip-flop che non controllano direttamente l'uscita, come nel caso delle porte logiche (gate), ma solo quando è permesso dal clock.[4]
Synchronous logic (Logico sincrono)	Il tipo di logica digitale usato in un sistema in cui le operazioni logiche si realizzano in sincronismo con impulsi di clock.
Synchronous operation (Funzionamento sincrono)	Funzionamento di un sistema sotto il controllo di impulsi di clock.[4]
Three-state device (Dispositivo three-state)	Un dispositivo logico a semiconduttore in cui esistono 3 possibili stati di uscita: (1) uno stato logico 0, (2) uno stato logico 1, (3) uno stato ad alta impedenza in cui l'uscita è, a tutti gli effetti, disconnessa dal resto del circuito, e quindi non lo influenza.
Two-phase clock (Clock a due fasi)	Un dispositivo di timing a due uscite, che fornisce due serie continue di impulsi di timing tra loro sincronizzate, con un singolo impulso di clock dalla seconda serie che sempre segue un singolo impulso di clock dalla prima serie. In funzione del genere di clock a due fasi, gli impulsi nella prima e seconda serie possono o non possono sovrapporsi l'un l'altro. L'8080A utilizza un clock a due fasi non sovrapponentesi (non overlapping).
Word (Parola)	Un gruppo di bit contigui occupanti una o più locazioni di memoria. Per il microprocessore 8080A, una parola è definita come un gruppo di otto bit contigui occupanti una singola locazione di memoria.
Word lenght (Lunghezza di parola)	Il numero di bit contigui trattati come unità e che, normalmente, sono posti in una o più locazioni di memoria. Una lunghezza di parola più grande implica una più elevata precisione ed istruzioni più complesse.

IL CHIP MICROPROCESSORE 8080

Il microprocessore 8080 è un circuito integrato LSI a 40 pin, contenente 16 linee di indirizzo, 8 linee dati, 10 linee di controllo, 4 collegamenti di alimentazione, e due ingressi di clock. La configurazione dei pin e lo schema a blocchi del chip, sono date nel-

le Figg. 2-1 e 2-2. Se non vi è familiare leggere i pin di un circuito integrato, sappiate che la numerazione parte dal pin 1 e procede in senso antiorario, a partire da un indice (index mark) ad un lato del chip, visto sempre dall'alto.

Quaranta pin sono senz'altro scomodi da trattare, per cui è meglio dividerli in categorie funzionali: alimentazioni, indirizzi, I/O e clock.

Alimentazione

pin 28	+ 12 V	(40 mA tipico)
pin 20	+ 5 V	(60 mA tipico)
pin 11	— 5 V	(0,01 mA tipico)
pin 2	massa	

Le tolleranze di tensione sono di ± 5% con riferimento al potenziale di massa. Qualunque sorgente in grado di dare ± 15 e + 5 V e sufficiente corrente, può essere adattata al microprocessore 8080, con opportuni regolatori di tensione.

Clock

Il microprocessore 8080 richiede un *clock a due fasi.* Ricordate che un *clock* può essere un qualsiasi dispositivo in grado di generare almeno un impulso di clock, oppure un dispositivo di timing in un sistema, in grado di fornire una serie continua di impulsi di timing. Un *clock a due fasi* è un dispositivo di timing a due uscite che fornisce due serie continue di impulsi di timing sincronizzati tra loro,

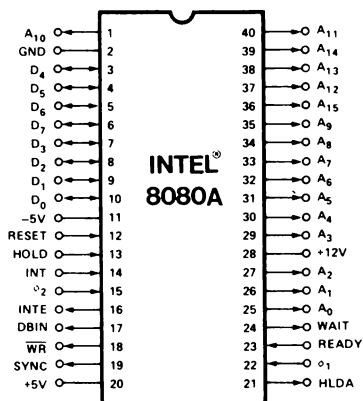



Fig. 2-1. Configurazione dei 40 pin del microprocessore 8080.

con un singolo impulso di clock dalla seconda serie che sempre segue un singolo impulso di clock dalla prima serie. L'uso

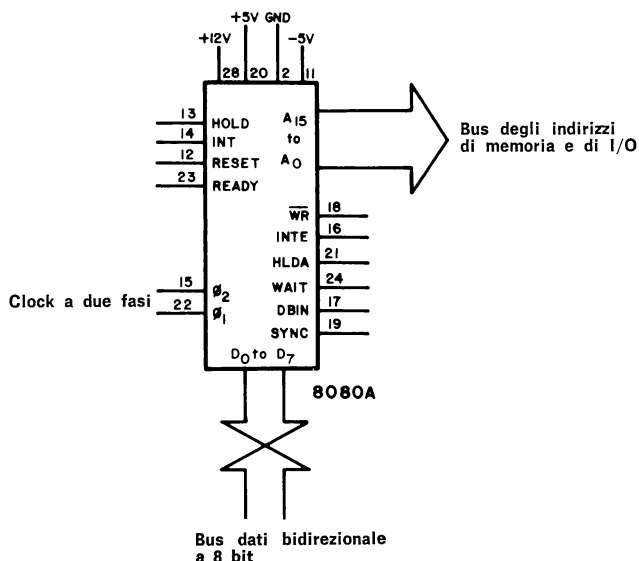



Fig. 2-2. Schema a blocchi del chip 8080 che mostra chiaramente il bus degli indirizzi a 16 bit ed il bus dati bidirezionale ad 8 bit. Questa è la rappresentazione più utile del microprocessore 8080.

dei diagrammi di tempo, come da Fig. 2-3, è utile per spiegare come opera un clock a due fasi. La frequenza può variare da 500 kHz a 4 MHz, in funzione del particolare chip 8080. La frequenza del clock non può essere ridotta a 0, a causa del fatto che le operazioni interne del microprocessore sono dinamiche e non statiche.

Notate che il fronte iniziale della serie $\varnothing_2$, quasi ricopre il fronte finale della serie $\varnothing_1$. Nelle specifiche dell'8080, la minima ampiezza per la fase $\varnothing_1$ è 60 ns., mentre per $\varnothing_2$ è 220 ns. I pin di ingresso del clock sono:

pin 22 fase $\varnothing_1$
pin 15 fase $\varnothing_2$

Chiamiamo questo tipo di clock come *clock a due fasi non sovrapponentesi.* Questo clock non è a livello TTL; piuttosto va da 0 a + 12 Volt. Tale clock può essere facilmente generato con un clock generator 8224, fornito dalla Intel o da altre case.

Indirizzi di Memoria

Il microprocessore 8080 può indirizzare *in modo diretto* fino ad un massimo di 65.536 parole di memoria ad 8 bit, tramite una serie di 16 linee di indirizzo three-state, detto *address bus.* I pin interessati sono i seguenti:

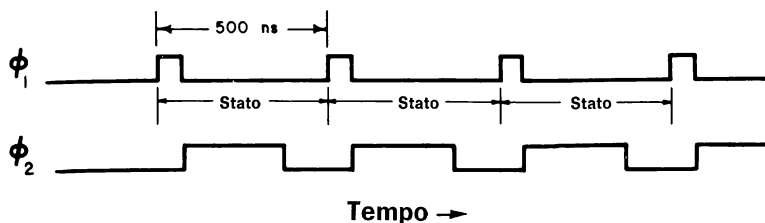



Fig. 2-3. Forme d'onda degli ingressi di clock a due fasi.

Pin	Byte	Funzione
pin 25	BYTE DI INDIRIZZO LO	Bit di indirizzo A_0, il bit meno significativo (LSB)
pin 26		Bit di indirizzo A_1
pin 27		Bit di indirizzo A_2
pin 29		Bit di indirizzo A_3
pin 30		Bit di indirizzo A_4
pin 31		Bit di indirizzo A_5
pin 32		Bit di indirizzo A_6
pin 33		Bit di indirizzo A_7, l'MSB nel byte LO
pin 34	BYTE DI INDIRIZZO HI	Bit di indirizzo A_8, l'LSB nel byte HI
pin 35		Bit di indirizzo A_9
pin 1		Bit di indirizzo A_{10}
pin 40		Bit di indirizzo A_{11}
pin 37		Bit di indirizzo A_{12}
pin 38		Bit di indirizzo A_{13}
pin 39		Bit di indirizzo A_{14}
pin 36		Bit di indirizzo A_{15}, il bit più significativo (MSB)

Sia i bit da A_0 ad A_7 e da A_8 fino ad A_{15}, possono essere usati per fornire indirizzi di dispositivi di I/O, fino ad un massimo di 256 di ingresso e 256 di uscita. Le linee di indirizzo sono inviate verso dei decoder, che forniscono la possibilità di selezionare un singolo dispositivo di I/O tra i 2^8 differenti possibili.

Bus Dati Bidirezionale

Il microprocessore 8080 è un dispositivo ad 8 bit, cioè vi è un accumulatore ad 8 bit, più registri addizionali ad 8 bit ed un *bus dati (data bus)* di I/O ad 8 bit. Questo bus è bidirezionale, cioè i dati possono uscire ed entrare dal chip. Il bus dati è la principale linea di comunicazione tra la CPU nel microprocessore, e la memoria, ed in genere il mondo esterno. Questo bus è un bus di I/O three-state. Le locazioni dei pin sono:

pin 10	Bit dati D_0, il bit meno significativo del bus dati
pin 9	Bit dati D_1
pin 8	Bit dati D_2
pin 7	Bit dati D_3
pin 3	Bit dati D_4
pin 4	Bit dati D_5
pin 5	Bit dati D_6
pin 6	Bit dati D_7, il bit più significativo del bus dati

Controlli

I pin di controllo determinano il funzionamento del microprocessore nell'ambito di un sistema a microcomputer. Nel discutere le funzioni di questo pin, è possibile prescindere da un certo numero di termini particolari, quali *T1, T2, T3, TW, fetch, ciclo, M1* ed altri. L'identificazione dei pin e le relative descrizioni sono fornite di seguito, per successivo riferimento quando sarete più esperti in questi argomenti.

Non vedrete, nella lista seguente, né un pin di controllo IN né un pin OUT. Il motivo è che queste due funzioni sono generate come *bit di stato,* esternamente sottoposte a latch ed utilizzate per generare gli impulsi di sincronizzazione $\overline{IN}$ e $\overline{OUT}$. Se vi interessa sapere subito come avviene, andate al capitolo 6.

I 4 pin di ingresso di controllo del microprocessore 8080 sono:

pin 21 (ingresso) RESET. Un 1 logico a questo ingresso, azzererà il program counter e permetterà al programma di partire dalla locazione di memoria HI = 000_8 e LO = 000_8. I flag INTE e HLDA sono resettati, ma i flag di condizione, l'accumulatore, lo stack pointer e gli altri registri, restano invariati.

pin 14 (ingresso) INT, o interrupt request. Un 1 logico a questo ingresso genererà una richiesta di interrupt che la CPU riconosce alla fine dell'istruzione corrente, oppure in stato di Alt. Se la CPU è nello stato HOLD, oppure se il flip-flop di interrupt enable è resettato allo stato logico 0, la richiesta non è presa in considerazione.

pin 23 (ingresso) READY. Un 1 logico indica al microprocessore 8080 che è disponibile, sul data bus, un dato valido, da DO a D7. Tale segnale, è utilizzato per sincronizzare la CPU con memorie più lente o con dispositivi di I/O. Se, dopo l'invio di un indirizzo sul bus, l'8080 non riceve un 1 all'ingresso READY, il microprocessore entra nello stato di WAIT finché READY resta ad 1. In tal modo è possibile realizzare il «single step».

pin 13 (ingresso) HOLD. Questo pin realizza l'ingresso in HOLD della CPU; in tale stato, un dispositivo esterno può prendere il controllo dei bus dati e indirizzi dell'8080. Questi bus, in HOLD, sono nello stato di alta impedenza. La CPU riconosce lo stato di HOLD con HLDA, o HOLD ACKNOWLEDGE, pin di uscita. HOLD è riconosciuto sotto due condizioni: (1) la CPU è nello stato di HALT, o (2) la CPU è negli stati T_2 o T_w ed il segnale READY è ad 1 logico.

Questo è quanto, per i segnali di controllo di ingresso. Ora vediamo i controlli in uscita; molti di questi sono flag. Il termine *flag* è stato definito precedentemente come:

Flag — In un computer, l'indicazione che una particolare operazione è stata completetata.[4] Un flag tipicamente è un flip-flop che può essere settato o azzerato (resettato, cleared) in risposta ad operazioni che si verificano nel microcomputer.

I sei pin di controllo in uscita dal microprocessore 8080 sono:

pin 24 (uscita) WAIT. L'uscita di WAIT, informa che la CPU è nello stato di WAIT, nel qual caso il pin è ad 1 logico.

pin 18 (uscita) $\overline{WR}$ o $\overline{WRITE}$. Questa uscita è usata per le operazioni di scrittura in memoria e di controllo di I/O. Quando il pin è a 0 logico, il dato sul bus dati è stabile, e può essere scritto in memoria o indirizzato verso un dispositivo periferico.

pin 21 (uscita) HLDA o HOLD ACKNOWLEDGE. Questo pin va ad 1 logico in risposta ad un input di HOLD. Indica che i bus dei dati e degli indirizzi sono nei loro stati di alta impedenza. Il segnale HLDA inizia a questi istanti: (1) al T_3 della lettura di memoria o di input, oppure (2) al periodo di clock che segue T_3, per operazioni di scrittura in memoria od operazioni di uscita.

pin 16 (uscita) INTE, INTERRUPT ENABLE. Questo pin indica lo stato del flip-flop di interrupt enable. Questo flip-flop può essere settato od azzerato delle istruzioni di abilitazione o disabilitazione interrupt (**373**$_8$ e **363**$_8$), e inibisce l'accettazione di segnali di interrupt da parte della CPU. Il flip-flop è automaticamente azzerato (da cui una disabilitazione di successivi interrupt) quando un interrupt è accettato. Il flip-flop è anche azzerato dall'ingresso di RESET.

pin 19 (uscita) SYNC, o SYNCHRONIZING SIGNAL. Il pin SYNC fornisce un 1 logico per indicare l'inizio di ogni ciclo macchina.

pin 17 (uscita) DBIN o DATA BUS IN. Quando questo pin va allo stato logico 1, indica alla circuiteria esterna che il data bus è nel «modo di ingresso». Il pin è usato per abilitare il gating dei dati nel bus dati del microprocessore 8080, dalla memoria o da dispositivi di I/O.

Alcune delle caratteristiche dei pin di controllo, vi diventeranno chiare nel Capitolo 6, dove $\overline{\text{WR}}$, SYNC, DBIN, READY, tutti pin di controllo, saranno discussi.

IL CLOCK GENERATOR/DRIVER 8224

Nei primi sistemi microcomputer 8080, gli ingressi di clock erano forniti da circuiti di driver o transistor, chip driver MOS oppure ancora buffer open collector TTL. Tutti lavoravano in modo adeguato, ma erano di non semplice progettazione. Un recente chip di interfaccia, il clock generator/driver 8224, contiene un oscillatore interno ed un clock generator/driver. Tutto quello che ci serve è un quarzo appropriato e le alimentazioni + 5 e + 12 V. Dato che il chip 8224 divide la frequenza del quarzo per nove, vi serve un cristallo a 18 MHz per avere il clock a 2 MHz, in uscita dal clock generator. Nel sistema descritto in questo capitolo, la frequenza del microcomputer è 750 kHz, per cui serve un quarzo da 6,750 MHz.

Le specifiche della Intel per il clock generator 8224, sono indicate nella loro forma originale nelle pagine seguenti. La descrizione funzionale del chip è eccellente, per cui si è ritenuto opportuno riportare esattamente le pagine del manuale Intel. Osservate come il circuito che determina la divisione per nove entro il chip è usata per generare le due fasi di clock $\varnothing_1$ e $\varnothing_2$ tra + 12 V e massa.

Gli ingressi e le uscite dal chip 8224, possono essere così riassunti:

pin 15, pin 14 XTAL 1 e XTAL 2. Il quarzo è connesso a questi due pin.

pin 13 TANK. Usato per attuare l'«overtone» di cristalli che hanno un guadagno molto più basso di quelli che operano alla frequenza fondamentale.

pin 2 (ingresso) $\overline{\text{RESIN}}$. Con l'ausilio di un circuito di trigger di Schmitt, interno al chip, e di una rete RC esterna, questo ingresso converte una lenta transizione nell'alimentazione, in un fronte pulito e rapido, che resetta il microprocessore 8080A, quan-

Schottky Bipolar 8224

THE CLOCK GENERATOR AND DRIVER FOR 8080A CPU

- **Single Chip Clock Generator/Driver for 8080A CPU**
- **Power-Up Reset for CPU**
- **Ready Synchronizing Flip-Flop**
- **Advanced Status Strobe**
- **Oscillator Output for External System Timing**
- **Crystal Controlled for Stable System Operation**
- **Reduces System Package Count**

The 8224 is a single chip clock generator/driver for the 8080A CPU. It is controlled by a crystal, selected by the designer, to meet a variety of system speed requirements.

Also included are circuits to provide power-up reset, advance status strobe and synchronization of ready.

The 8224 provides the designer with a significant reduction of packages used to generate clocks and timing for 8080A.

PIN CONFIGURATION

BLOCK DIAGRAM

PIN NAMES

Pin	Description	Pin	Description
RESIN	RESET INPUT	XTAL 1	CONNECTIONS FOR CRYSTAL
RESET	RESET OUTPUT	XTAL 2	
RDYIN	READY INPUT	TANK	USED WITH OVERTONE XTAL
READY	READY OUTPUT	OSC	OSCILLATOR OUTPUT
SYNC	SYNC INPUT	ϕ_2 (TTL)	ϕ_2 CLK (TTL LEVEL)
STSTB	STATUS STB (ACTIVE LOW)	V_{CC}	+5V
ϕ_1	8080 CLOCKS	V_{DD}	+12V
ϕ_2		GND	0V

SCHOTTKY BIPOLAR 8224

FUNCTIONAL DESCRIPTION

General

The 8224 is a single chip Clock Generator/Driver for the 8080A CPU. It contains a crystal-controlled oscillator, a "divide by nine" counter, two high-level drivers and several auxiliary logic functions.

Oscillator

The oscillator circuit derives its basic operating frequency from an external, series resonant, fundamental mode crystal. Two inputs are provided for the crystal connections (XTAL1, XTAL2).

The selection of the external crystal frequency depends mainly on the speed at which the 8080A is to be run at. Basically, the oscillator operates at 9 times the desired processor speed.

A simple formula to guide the crystal selection is:

$$\text{Crystal Frequency} = \frac{1}{t_{CY}} \text{ times } 9$$

Example 1: (500ns t_{CY})
2mHz times 9 = 18mHz*

Example 2: (800ns t_{CY})
1.25mHz times 9 = 11.25mHz

Another input to the oscillator is TANK. This input allows the use overtone mode crystals. This type of crystal generally has much lower "gain" than the fundamental type so an external LC network is necessary to provide the additional "gain" for proper oscillator operation. The external LC network is connected to the TANK input and is AC coupled to ground. See Figure 4.

The formula for the LC network is:

$$F = \frac{1}{2\pi\sqrt{LC}}$$

The output of the oscillator is buffered and brought out on OSC (pin 12) so that other system timing signals can be derived from this stable, crystal-controlled source.

*When using crystals above 10mHz a small amount of frequency "trimming" may be necessary. The addition of a small capacitance (3pF - 10pF) in series with the crystal will accomplish this function.

Clock Generator

The Clock Generator consists of a synchronous "divide by nine" counter and the associated decode gating to create the waveforms of the two 8080A clocks and auxiliary timing signals.

The waveforms generated by the decode gating follow a simple 2-5-2 digital pattern. See Figure 2. The clocks generated; phase 1 and phase 2, can best be thought of as consisting of "units" based on the oscillator frequency. Assume that one "unit" equals the period of the oscillator frequency. By multiplying the number of "units" that are contained in a pulse width or delay, times the period of the oscillator frequency, the approximate time in nanoseconds can be derived.

The outputs of the clock generator are connected to two high level drivers for direct interface to the 8080A CPU. A TTL level phase 2 is also brought out ϕ_2 (TTL) for external timing purposes. It is especially useful in DMA dependant activities. This signal is used to gate the requesting device onto the bus once the 8080A CPU issues the Hold Acknowledgement (HLDA).

Several other signals are also generated internally so that optimum timing of the auxiliary flip-flops and status strobe ($\overline{STSTB}$) is achieved.

SCHOTTKY BIPOLAR 8224

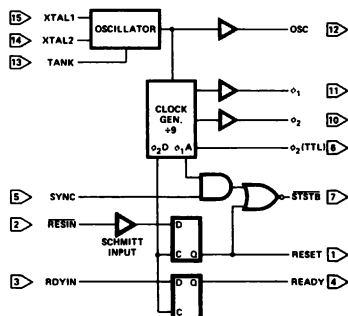
$\overline{STSTB}$ (Status Strobe)

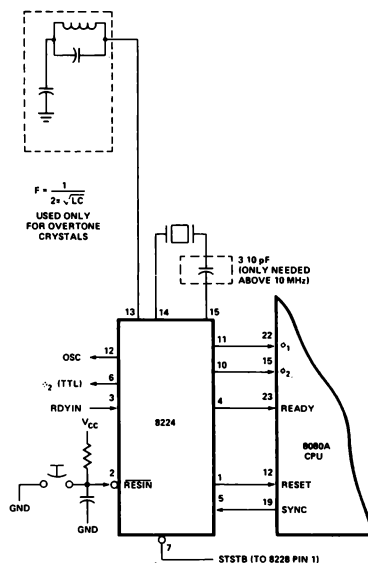
At the beginning of each machine cycle the 8080A CPU issues status information on its data bus. This information tells what type of action will take place during that machine cycle. By bringing in the SYNC signal from the CPU, and gating it with an internal timing signal (ϕ1A), an active low strobe can be derived that occurs at the start of each machine cycle at the earliest possible moment that status data is stable on the bus. The $\overline{STSTB}$ signal connects directly to the 8228 System Controller.

The power-on Reset also generates $\overline{STSTB}$, but of course, for a longer period of time. This feature allows the 8228 to be automatically reset without additional pins devoted for this function.

Power-On Reset and Ready Flip-Flops

A common function in 8080A Microcomputer systems is the generation of an automatic system reset and start-up upon initial power-on. The 8224 has a built in feature to accomplish this feature.

An external RC network is connected to the $\overline{RESIN}$ input. The slow transition of the power supply rise is sensed by an internal Schmitt Trigger. This circuit converts the slow transition into a clean, fast edge when its input level reaches a predetermined value. The output of the Schmitt Trigger is connected to a "D" type flip-flop that is clocked with ϕ2D (an internal timing signal). The flip-flop is synchronously reset and an active high level that complies with the 8080A input spec is generated. For manual switch type system Reset circuits, an active low switch closing can be connected to the $\overline{RESIN}$ input in addition to the power-on RC net-network.

The READY input to the 8080A CPU has certain timing specifications such as "set-up and hold" thus, an external synchronizing flip-flop is required. The 8224 has this feature built-in. The RDYIN input presents the asynchronous "wait request" to the "D" type flip-flop. By clocking the flip-flop with ϕ2D, a synchronized READY signal at the correct input level, can be connected directly to the 8080A.

The reason for requiring an external flip-flop to synchronize the "wait request" rather than internally in the 8080 CPU is that due to the relatively long delays of MOS logic such an implementation would "rob" the designer of about 200ns during the time his logic is determining if a "wait" is necessary. An external bipolar circuit built into the clock generator eliminates most of this delay and has no effect on component count.

do l'uscita RESET è connessa all'8080A. Uno switch di reset manuale può essere anche connesso a $\overline{\text{RESIN}}$.

pin 1 (uscita) RESET. Un'uscita a zero logico che è applicato all'ingresso RESET del microprocessore 8080 al fine di resettarlo.

pin 3 (ingresso) RDYIN. Accetta una richiesta asincrona di attesa e la sincronizza per produrre un segnale READY posto in uscita.

pin 4 (uscita) READY. Uno stato logico 1 indica al microprocessore 8080 che sul bus dei dati sono disponibili dati validi dalla memoria o da dispositivi esterni.

pin 5 (ingresso) SYNC. Il pin SYNC fornisce una uscita di sincronizzazione verso il chip 8224, al fine di indicare l'inizio di ogni ciclo macchina.

pin 11 (uscita) $\varnothing_1$ e $\varnothing_2$. Clock a due fasi posto in uscita all'8080.

pin 10 (uscita) Ciascuna delle due uscite varia tra + 12 e massa; non sono delle normali uscite TTL.

pin 6 (uscita) $\varnothing_2$ (TTL). Questa è una uscita di clock TTL, con le stesse caratteristiche e timing di $\varnothing_2$. Viene utilizzato per temporizzazioni esterne, come descritto nel Capitolo 5.

pin 7 (uscita) $\overline{\text{STSTB}}$. Uscita Status Strobe. Questa uscita è usata per il latch dei bit di stato che compaiono sul bus dei dati bidirezionale.

pin 12 (uscita) OSC. Uscita bufferizzata dell'oscillatore al quarzo, usabile per realizzare timing esterni.

Dovrebbe essere chiaro che il chip clock generator/driver 8224 è ben progettato per le sue specifiche funzioni; le connessioni tra esso e l'8080 sono dirette, senza l'intermediazione di inverter, gate o flip-flop. Non vi è quindi una particolare motivazione ad utilizzare circuiti di driver a transistor, clock driver MOS o buffer TTL open collector. L'alimentazione per il chip 8224 è poi già disponibile, in quanto il + 5 ed il + 12 sono comunque richieste dal microprocessore 8080.

UN MICROCOMPUTER BASATO SUL MICROPROCESSORE 8080

La Fig. 2-4 mostra la sezione del processore centrale di un piccolo microcomputer basato sull'8080A. La figura è presa dalla rivista *Radio Electronics*, che ha descritto il microcomputer nei mesi di maggio, giugno e luglio 1976. Noi ora esamineremo i chip

Fig. 2-4. La sezione processore, memoria e controllo del piccolo sistema.

costituenti il circuito ed il flusso dei segnali tra di essi. L'obiettivo è dimostrare che un microcomputer è un dispositivo ragionevole e lineare, per cui non deve assolutamente intimidirci.

Alimentazione

Si ipotizza che siano disponibili gli alimentatori per + 5,

—12, + 12 V; del resto sono di facile reperibilità e basso costo. Le tensioni intermedie —5 e —9 V sono ottenibili con l'uso di regolatori integrati, come la serie LM320, oppure di diodi zener, come nella Fig. 2-5.

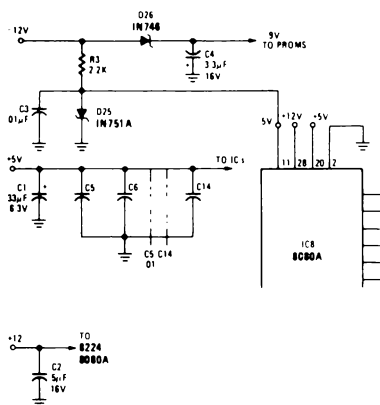



Fig. 2-5. L'uso dei diodi zener fornisce le tensioni —5 e —9 V necessarie al microcomputer. Si suppone che + 5 + 12 e —12 V siano disponibili da un alimentatore.

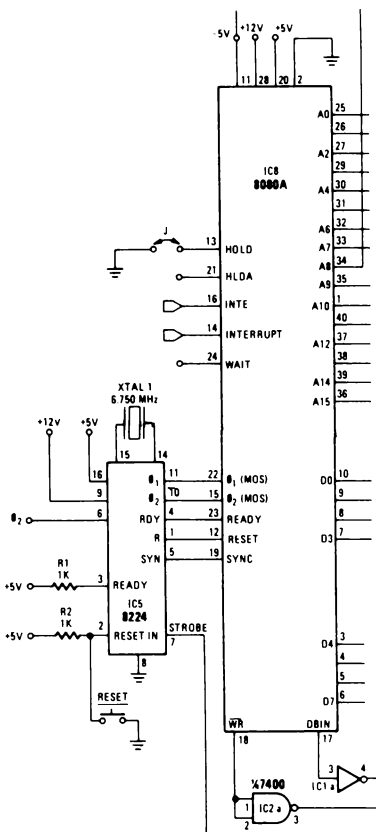
Il Chip Microprocessore 8080A

I pin di uscita dell'8080 hanno un fan-out di un ingresso TTL low-power, circa 0,16 mA. Le specifiche di uscita per il chip 8080A sono di 1,9 mA per ogni pin di uscita, quindi un fan-out di poco superiore ad un carico TTL standard. Nessuno dei fan-out visti può essere considerato però come buono. Quindi è necessario ricorrere ai bus driver, che saranno fra poco descritti.

Linee di Controllo

La sezione di controllo è mostrata nella Fig. 2-6. E' presente il clock generator/driver 8224 precedentemente descritto, connesso direttamente all'8080A. I soli componenti addizionali richiesti sono una coppia di resistenze da 1 kΩ, uno switch di reset ed un quarzo da 6,750 MHz.

Le restanti linee di controllo sul chip 8080A, quelle non connesse all'8224, sono: HOLD, HLDA, INTE, INTERRUPT, WAIT, $\overline{\text{WR}}$ e DBIN. Cinque di queste linee non sono usate nel microcomputer, ma sono disponibili nel caso desiderate sperimentarle. L'input di HOLD vi permette di pilotare il chip 8080A nello stato di hold e di disabilitare i bus dati ed indirizzi. L'ingresso INTERRUPT vi permette di interrompere l'esecuzione di un programma, facendo si che il flip-flop di interrupt entro il chip, sia abilitato. Se è abilitato, l'uscita INTE è allo stato logico 1. L'uscita di controllo HLDA «sente» l'esistenza dello stato di hold. Ed infine l'u-

Fig. 2-6. Il controllo del chip 8080 è realizzato tramite l'uso di un clock generator/driver 8224 e di un quarzo a 6,750 MHz.

scita WAIT permette all'8080 di segnalare che non è pronto, oppure che è in attesa di qualche evento esterno. Se l'ingresso HOLD non è usato, deve essere posto a massa, e questo è facilmente ottenibile con un ponticello, come mostrato nella Fig. 2-6.

Le due ultime linee di controllo, sono di uscita. Il segnale WRITE ($\overline{WR}$) è attivo quando è allo stato 0 logico, ed indica che l'8080A sta inviando dati verso l'esterno. Il segnale DBIN (data bus in), indica che il bus dati sta per essere usato per l'input di dati. E' attivo allo stato logico 1. Nell'8080A, il bus dati è bidirezionale, cioè il trasferimento dati fuori dal chip o verso il chip, si realizza sulle stesse vie di collegamento. E' quindi necessaria una attenta gestione di questa linea, per un corretto realizzarsi dei trasferimenti dati. La possibilità di un corretto «data bus management» è insita del resto, entro il chip stesso, ma comunque dobbiamo essere certi che i dispositivi esterni non cerchino di porre i loro

dati sul bus contemporaneamente ad altri dispositivi od al chip 8080A stesso. Solo un dispositivo alla volta può trasmettere i dati sul bus dati, ad ogni istante di tempo.

Bus Driver

Al fine di pilotare i chip di memoria ed i latch di uscita nel nostro piccolo microcomputer 8080, è richiesto un fan-out di almeno 10, per ogni linea di uscita sul data bus. In più occorre mantenere il carattere bidirezionale del dato. Il dispositivo da utilizzarsi è il bus driver bidirezionale parallelo a 4 bit 8216 della Intel, di cui si danno, nelle pagine successive, le caratteristiche originali fornite dalla Intel Corporation. Considerate l'uscita DB_O, nel diagramma logico del chip 8216. Valgono le seguenti tabelle della verità:

$\overline{DIEN}$	$\overline{CS}$	
0	0	$DI_0 \rightarrow DB_0$, il dato esce dall'8080
1	0	$DB_0 \rightarrow DO_0$, il dato entra nell'8080
0	1	Stato di alta impedenza; chip disabilitati
1	1	Stato di alta impedenza; chip disabilitati

In altre parole, quando $\overline{DIEN}$ è a 0 logico ed il chip è abilitato, l'8216 agisce da buffer di ingresso. Quando $\overline{DIEN}$ è ad 1 logico ed il chip è abilitato, l'8216 agisce da buffer di uscita.

La sezione bus driver del nostro microcomputer, è mostrata nella Fig. 2-7. Osservate che DBIN è connesso a $\overline{DIEN}$ (pin 15 del chip 8216) e che ogni 8216 è permanentemente abilitato. La tabella della verità relativa a DBIN e $\overline{DIEN}$ è:

DBIN	$\overline{DIEN}$	
0	0	Il dato esce dall'8080: il bus dati è in output mode
1	1	Il dato entra nell'8080: il bus dati è in input mode

In base alle specifiche Intel per l'8216, la corrente massima assoluta in uscita a 0 logico è 125 mA. Notate che sia DBIN che $\overline{WR}$ sono sottoposti a buffer da un 7400 o da un 7404, per portare il loro fan-out da uno a dieci carichi standard TTL.

Informazioni di Stato

Studiando attentamente le specifiche Intel dell'8080, osserverete che certi importanti segnali di controllo non sono presenti sul chip stesso. Fra questi segnali vi è il memory read $(\overline{MR})$, memory write $(\overline{MW})$, input $(\overline{IN})$, output $(\overline{OUT})$, ed interrupt acknowledge $(\overline{INTA})$. Per generare tali segnali, l'8080 usa una tecnica

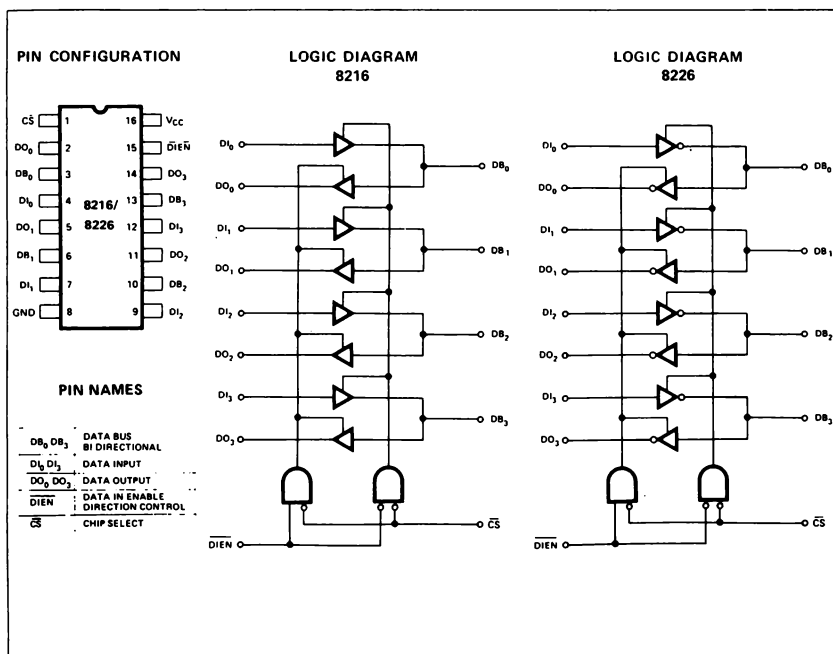
intel® Schottky Bipolar 8216/8226

4 BIT PARALLEL BIDIRECTIONAL BUS DRIVER

- **Data Bus Buffer Driver for 8080 CPU**
- **Low Input Load Current — .25 mA Maximum**
- **High Output Drive Capability for Driving System Data Bus**
- **3.65V Output High Voltage for Direct Interface to 8080 CPU**
- **Three State Outputs**
- **Reduces System Package Count**

The 8216/8226 is a 4-bit bi-directional bus driver/receiver.

All inputs are low power TTL compatible. For driving MOS, the DO outputs provide a high 3.65V V_{OH}, and for high capacitance terminated bus structures, the DB outputs provide a high 50mA I_{OL} capability.

A non-inverting (8216) and an inverting (8226) are available to meet a wide variety of applications for buffering in microcomputer systems.

SCHOTTKY BIPOLAR 8216/8226

FUNCTIONAL DESCRIPTION

Microprocessors like the 8080 are MOS devices and are generally capable of driving a single TTL load. The same is true for MOS memory devices. While this tvpe of drive is sufficient in small systems with few components, quite often it is necessary to buffer the microprocessor and memories when adding components or expanding to a multi-board system.

The 8216/8226 is a four bit bi-directional bus driver specifically designed to buffer microcomputer system components.

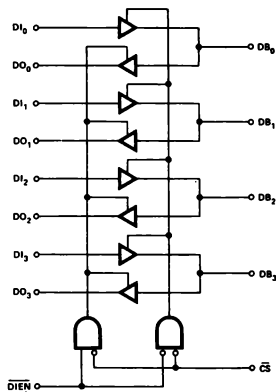
Bi-Directional Driver

Each buffered line of the four bit driver consists of two separate buffers that are tri-state in nature to achieve direct bus interface and bi-directional capability. On one side of the driver the output of one buffer and the input of another are tied together (DB), this side is used to interface to the system side components such as memories, I/O, etc., because its interface is direct TTL compatible and it has high drive (50mA). On the other side of the driver the inputs and outputs are separated to provide maximum flexibility. Of course, they can be tied together so that the driver can be used to buffer a true bi-directional bus such as the 8080 Data Bus. The DO outputs on this side of the driver have a special high voltage output drive capability (3.65V) so that direct interface to the 8080 and 8008 CPUs is achieved with an adequate amount of noise immunity (350mV worst case).

Control Gating $\overline{\text{DIEN}}$, $\overline{\text{CS}}$

The $\overline{\text{CS}}$ input is actually a device select. When it is "high" the output drivers are all forced to their high-impedance state. When it is at "zero" the device is selected (enabled) and the direction of the data flow is determined by the $\overline{\text{DIEN}}$ input.

The $\overline{\text{DIEN}}$ input controls the direction of data flow (see Figure 1) for complete truth table. This direction control is accomplished by forcing one of the pair of buffers into its high impedance state and allowing the other to transmit its data. A simple two gate circuit is used for this function.

The 8216/8226 is a device that will reduce component count in microcomputer systems and at the same time enhance noise immunity to assure reliable, high performance operation.

(a) 8216

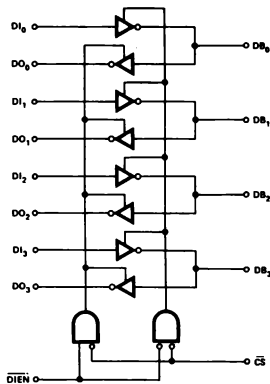
(b) 8226

$\overline{\text{DIEN}}$	$\overline{\text{CS}}$	
0	0	DI → DB
1	0	DB → DO
0	1	HIGH IMPEDANCE
1	1	HIGH IMPEDANCE

Figure 1. 8216/8226 Logic Diagrams

SCHOTTKY BIPOLAR 8216/8226

APPLICATIONS OF 8216/8226

8080 Data Bus Buffer

The 8080 CPU Data Bus is capable of driving a single TTL load and is more than adequate for small, single board systems. When expanding such a system to more than one board to increase I/O or Memory size, it is necessary to provide a buffer. The 8216/8226 is a device that is exactly fitted to this application.

Shown in Figure 2 are a pair of 8216/8226 connected directly to the 8080 Data Bus and associated control signals. The buffer is bi-directional in nature and serves to isolate the CPU data bus.

On the system side, the DB lines interface with standard semiconductor I/O and Memory components and are completely TTL compatible. The DB lines also provide a high drive capability (50mA) so that an extremely large system can be dirven along with possible bus termination networks.

On the 8080 side the DI and DO lines are tied together and are directly connected to the 8080 Data Bus for bi-directional operation. The DO outputs of the 8216/8226 have a high voltage output capability of 3.65 volts which allows direct connection to the 8080 whose minimum input voltage is 3.3 volts. It also gives a very adequate noise margin of 350mV (worst case).

The control inputs to 8216/8226 ($\overline{CS}$, $\overline{DIEN}$) are connected directly to the 8080. $\overline{DIEN}$ is tied to DBIN so that proper bus flow is maintained, and $\overline{CS}$ is tied to HLDA so that the system side Data Bus will be 3-stated when a Hold request has been acknowledged during a DMA activity.

Memory and I/O Interface to a Bi-directional Bus

In large microcomputer systems it is often necessary to provide Memory and I/O with their own buffers and at the same time maintain a direct, common interface to a bi-directional Data Bus. The 8216/8226 has separated data in and data out lines on one side and a common bi-directional set on the other to accomodate such a function.

Shown in Figure 3 is an example of how the 8216/8226 is used in this type of application.

The interface to Memory is simple and direct. The memories used are typically Intel® 8102, 8102A, 8101 or 8107A and have separate data inputs and outputs. The DI and DO lines of the 8216/8226 tie to them directly and under control of the $\overline{MEMR}$ signal, which is connected to the $\overline{DIEN}$ input, an interface to the bi-directional Data Bus is maintained.

The interface to I/O is similar to Memory. The I/O devices used are typically Intel® 8255s, and can be used for both input and output ports. The $\overline{I/O\ R}$ signal is connected directly to the $\overline{DIEN}$ input so that proper data flow from the I/O device to the Data Bus is maintained.

The 8216/8226 can be used in a wide variety of other buffering functions in microcomputer systems such as Address Bus Drivers, Drivers to peripheral devices such as printers, and as Drivers for long length cables to other peripherals or systems.

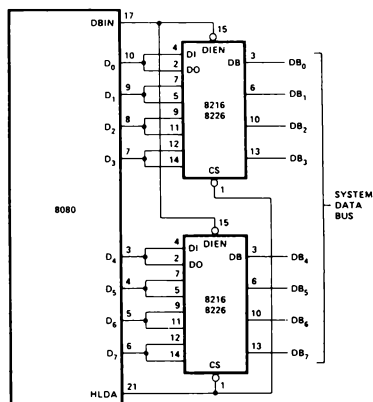



Figure 2. 8080 Data Bus Buffer.

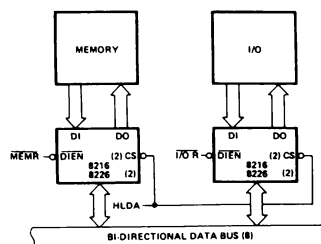



Figure 3. Memory and I/O Interface to a Bi-Directional Bus.

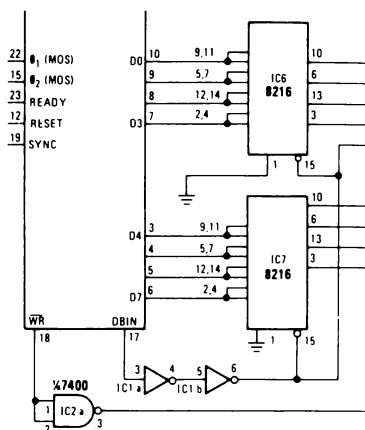



Fig. 2-7. Sezione bus driver di un piccolo microcomputer basato sull'8080.

«look ahead». Dato che il bus dati non è utilizzato per tutto il tempo in cui avviene il trasferimento, l'8080 può usare il bus per trascrivere un'ulteriore informazione di controllo. *Tale informazione è posta in uscita molto presto nel ciclo macchina* (vedere Capitolo 6) *per generare segnali di controllo utilizzati per controllare il trasferimento dati verso o da dispositivi di input/output e memoria.*

L'informazione di stato appare sul bus dati in un periodo molto breve, circa 500 ns per un microprocessore 8080 operante a 2 MHz. Dato che l'informazione deve essere usata in seguito, deve essere sottoposta a latch. Il SYSTEM STROBE ($\overline{STSTB}$) è generato al pin 7 dal chip 8224 nel giusto istante in cui l'informazione di stato deve essere catturata. Notate che il segnale $\overline{STSTB}$ è generato a partire dal clock $\varnothing_1$ e dal segnale SYNC dell'8080A. E' possibile usare qualsiasi tipo di latch. Nel Capitolo 6 verrà descritto l'uso del buffer/latch 8212. Nella Fig. 2-8 si è utilizzato un chip 74174 6-bit positive-edge-triggered latch, con clock al pin 9.

Tutti gli 8 bit sul bus dati forniscono una specie di informazione di stato, ma non tutti gli 8 bit sono necessari. Nelle pagine seguenti sono date tutte le informazioni originali Intel relative ai singoli bit di stato.

Nel nostro piccolo sistema 8080, i segnali di stato $\overline{WO}$ e STACK sono ignorati, in quanto a noi non molto utili. HLTA ed M1 sono sottoposti a latch, ma non usati. I segnali importanti sono INTA (interrupt acknowledge), INP (input), OUT (output), e MEMR (memory read). Insieme a DBIN e $\overline{WR}$, dall'8080, questi 4 segnali danno 5 controlli molto importanti, che costituiscono il nostro control bus:

- $\overline{MR}$. *Memory read.* Usato per realizzare lo strobe dei dati dalla memoria verso l'8080.
- $\overline{MW}$. *Memory write.* Usato per realizzare lo strobe dei dati dall'8080 verso una memoria esterna.
- $\overline{IN}$. *Input.* Usato per realizzare lo strobe dei dati da un dispositivo di input, nell'accumulatore dell'8080.
- $\overline{OUT}$. *Output.* Usato per realizzare lo strobe dell'accumulatore verso un dispositivo di input esterno all'8080.
- $\overline{INTA}$. *Interrupt acknowledge.* Usato per realizzare lo strobe di un singolo byte di istruzione nel registro istruzione entro l'8080 durante un interrupt.

Gli altri segnali sono: RESET, INT (interrupt) e INTE (interrupt enable).

Vi è, per attuare il buffer dati in modo bidirezionale ed il latch ed il gate dei segnali di stato, il system controller e bus driver 8228. Un tipico circuito di interfaccia è dato nelle pagine seguenti, per concessione della Intel Corporation. Il problema connesso all'utilizzo dell'8228 è fondamentalmente quello del suo costo; inoltre non fornisce l'accesso a tutti gli 8 status bit. Gli autori quindi preferiscono i chip 8216, 7400 e 74174.

Memoria

La necessaria sezione di controllo per l'8080A è a posto e i bit di stato sono sottoposti a latch. Siamo ora pronti ad aggiungere della circuiteria esterna al chip. I primi dispositivi che ci servono sono dei chip di memoria. Vi sono differenti tipi di memoria, ma noi ne consideriamo solo due, entrambe random access memory. *«Random Access»* indica che qualsiasi singola locazione di memoria può essere raggiunta a partire da una qualunque altra locazione. Come sottogruppo abbiamo le memorie lettura/scrittura (R/W) e le read only memory (ROM).

Noi sceglieremo le memorie R/W 2111, che sono del resto facili da interfacciare con il microprocessore 8080A. Queste memorie sono organizzate in 256 per 4, cioè 1024 bit oppure ancora, memoria da un kilobit. Le specifiche originali della Intel Corporation per l'8111-2, pin compatibili con la 2111, sono date nelle pagine seguenti. Il chip 8111-2 possiede delle linee comuni di ingresso ed uscita (I/O); in tali linee avviene il trasferimento dati da e per l'8080A. Chiaramente, queste linee di I/O sono bidirezionali. Ogni chip 8111-2 ha otto ingressi di indirizzo (A0-A7) per identificare in modo unico una fra le possibili 256 locazioni di memoria. Gli ingressi di controllo all'8111-2 sono: ingresso di lettura/scrittura (R/W), ingresso di chip enable ($\overline{CE_1}$ e $\overline{CE_2}$) ed un ingresso di output disable (OD).

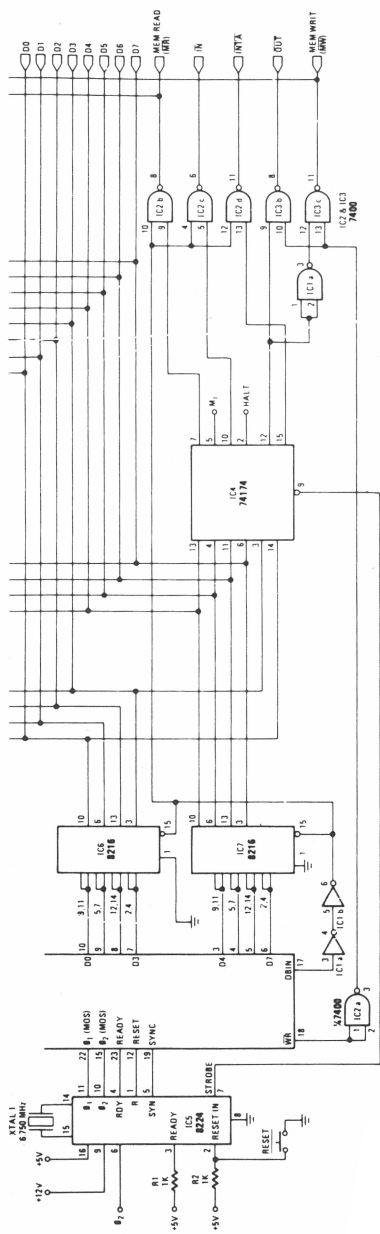
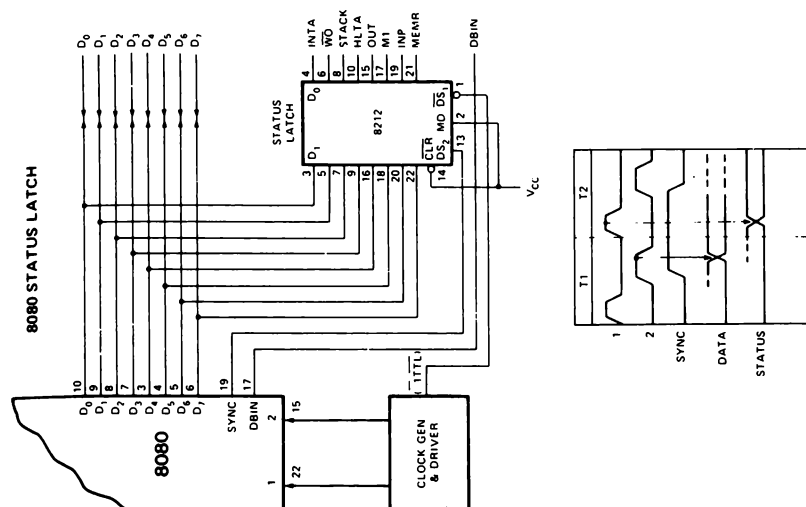
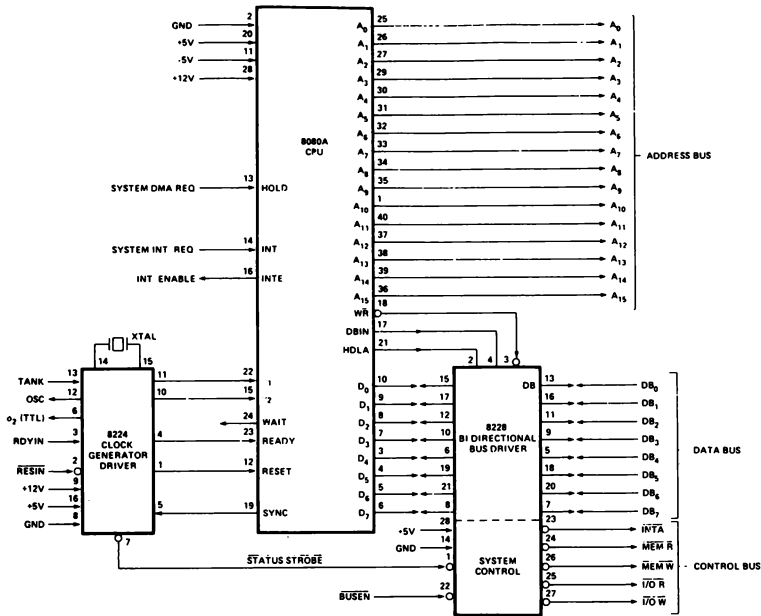



Fig. 2-8. La sezione controllo ed informazione di stato del piccolo sistema 8080. Lo status latch è il chip 74174, latch a 6 bit.

Instructions for the 8080 require from one to five machine cycles for complete execution. The 8080 sends out 8 bit of status information on the data bus at the beginning of each machine cycle (during SYNC time). The following table defines the status information.

STATUS INFORMATION DEFINITION

Symbols	Data Bus Bit	Definition
INTA*	D_0	Acknowledge signal for INTERRUPT request. Signal should be used to gate a restart instruction onto the data bus when DBIN is active.
$\overline{WO}$	D_1	Indicates that the operation in the current machine cycle will be a WRITE memory or OUTPUT function ($\overline{WO}$ = 0). Otherwise, a READ memory or INPUT operation will be executed.
STACK	D_2	Indicates that the address bus holds the pushdown stack address from the Stack Pointer.
HLTA	D_3	Acknowledge signal for HALT instruction.
OUT	D_4	Indicates that the address bus contains the address of an output device and the data bus will contain the output data when $\overline{WR}$ is active.
M_1	D_5	Provides a signal to indicate that the CPU is in the fetch cycle for the first byte of an instruction.
INP*	D_6	Indicates that the address bus contains the address of an input device and the input data should be placed on the data bus when DBIN is active.
MEMR*	D_7	Designates that the data bus will be used for memory read data.

8080 STATUS LATCH

Dato che la lunghezza della parola in ogni 8111-2 è solo 4 bit, coppie di tali chip devono essere abilitati in modo simultaneo per realizzare le parole di 8 bit richieste dal microprocessore 8080.

La Fig. 2-4 mostra che $\overline{MW}$ (memory write) è connesso a R/W (pin 16) sul chip 8111-2, e che $\overline{MR}$ (memory read) è connessa al pin 9 (OD) sul chip di lettura/scrittura.

Partendo dalla ipotesi che il chip è abilitato, la tabella della verità è la seguente:

$\overline{MW}$	$\overline{MR}$	R/W	OD	
0	0	[Nota: Combinazione di ingresso non possibile]		
0	1	0	1	Memory write, disable memory output
1	0	1	0	Memory read
1	1	1	1	Disable memory output

La decodifica degli indirizzi è ottenibile dalla Fig. 2-9. La desiderata tabella della verità è:

A15	*A14*	*A13*	*A12*	*A11*	*A10*	*A9*	*A8*	*A7 . . . A0*	*Memoria*	*Uso*
0	0	0	0	0	0	0	0	X . . . X	Blocco 0	Riservato per EPROM
0	0	0	0	0	0	0	1	X . . . X	Blocco 1	Riservato per EPROM
0	0	0	0	0	0	1	0	X . . . X	Blocco 2	8111 read/write memory
0	0	0	0	0	0	1	1	X . . . X	Blocco 3	8111 read/write memory

Silicon Gate MOS 8111-2

1024 BIT (256 x 4) STATIC MOS RAM WITH COMMON I/O AND OUTPUT DISABLE

- **Organization 256 Words by 4 Bits**
- **Access Time — 850 nsec Max.**
- **Common Data Input and Output**
- **Single +5V Supply Voltage**
- **Directly TTL Compatible — All Inputs and Output**
- **Static MOS — No Clocks or Refreshing Required**
- **Simple Memory Expansion — Chip Enable Input**
- **Fully Decoded — On Chip Address Decode**
- **Inputs Protected — All Inputs Have Protection Against Static Charge**
- **Low Cost Packaging — 18 Pin Plastic Dual-In-Line Configuration**
- **Low Power — Typically 150 mW**
- **Three-State Output — OR-Tie Capability**

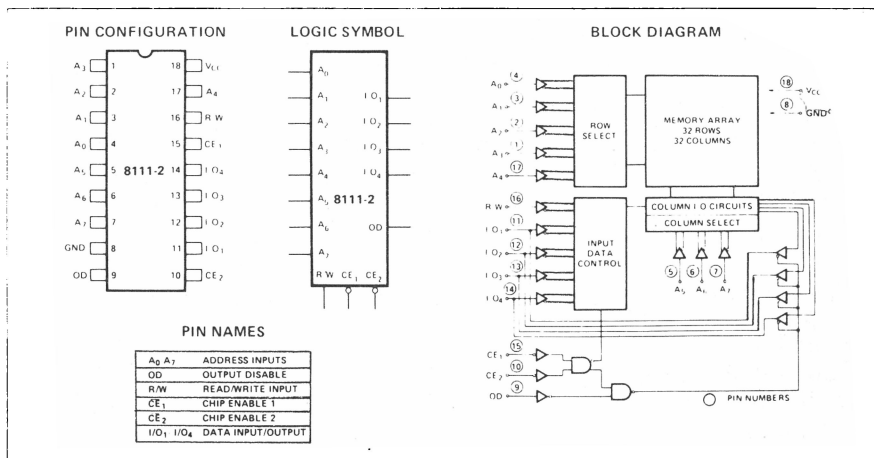
The Intel® 8111-2 is a 256 word by 4 bit static random access memory element using normally off N-channel MOS devices integrated on a monolithic array. It uses fully DC stable (static) circuitry and therefore requires no clocks or refreshing to operate. The data is read out nondestructively and has the same polarity as the input data. Common input/output pins are provided.

The 8111-2 is designed for memory applications in small systems where high performance, low cost, large bit storage, and simple interfacing are important design objectives.

It is directly TTL compatible in all respects: inputs, outputs, and a single +5V supply. Separate chip enable ($\overline{CE}$) leads allow easy selection of an individual package when outputs are OR-tied.

The Intel® 8111-2 is fabricated with N-channel silicon gate technology. This technology allows the design and production of high performance, easy-to-use MOS circuits and provides a higher functional density on a monolithic chip than either conventional MOS technology or P-channel silicon gate technology.

Intel's silicon gate technology also provides excellent protection against contamination. This permits the use of low cost silicone packaging.

A_0–A_7	ADDRESS INPUTS
OD	OUTPUT DISABLE
R/W	READ/WRITE INPUT
$\overline{CE}_1$	CHIP ENABLE 1
CE_2	CHIP ENABLE 2
I/O_1–I/O_4	DATA INPUT/OUTPUT

Qui una X indica che è possibile sia uno 0 che un 1 logico. I bit da A0 ad A7 possono essere una qualsiasi combinazione di stati logici 0 ed 1, per un totale di 256 differenti combinazioni. Osservate che solo i bit di indirizzo A8 ed A9 cambiano dando tutte le possibili combinazioni per due bit.

I bit di indirizzo da A10 ad A15, restano a 0 logico per tutti i nostri selezionati indirizzi nel sistema microcomputer.

E' pratica comune decodificare in assoluto le locazioni di memoria, cioè, assicurare che tutti i 16 bit di indirizzo partecipino nella decodifica di una locazione di memoria fornendo l'ingresso chip enable ($\overline{CE}$) insieme agli otto indirizzi da A0 ad A7. La Fig. 2-9 dimostra come ciò è realizzato. Dato che i bit di indirizzo da A10 ad A15 restano a 0, usiamo gli inverter open-collector in una configurazione wired-or per dare una unica condizione logica decodificata. Osservate la presenza della resistenza di pull-up da 1 kΩ, R4. La tabella della verità per il circuito è la seguente:

A15	*A14*	*A13*	*A12*	*A11*	*A10*	*Q*
0	0	0	0	0	0	1
X	X	X	X	X	1	0
X	X	X	X	1	X	0
X	X	X	1	X	X	0
X	X	1	X	X	X	0
X	1	X	X	X	X	0
1	X	X	X	X	X	0

Nota: X = 0 zero logico od 1 logico

Osservate che questa tabella della verità è identica a quella ottenibile da una porta NOR a 6 ingressi; lo stato logico unico è Q = 1, e tale condizione di uscita si realizza solo quando tutti gli ingressi sono a 0 logico.

Ogni qualvolta l'uscita del wired-or, o del «six-input NOR», è ad 1, sappiamo che da A10 ad A15 è presente 0 logico, e quindi siamo entro uno dei quattro blocchi di memoria a 256 byte. Dobbiamo poi restringere il processo di selezione della memoria verso un blocco specifico. Ciò è fatto tramite un decoder 74LS155, di cui diamo le specifiche nelle Figg. 2-9 e 2-10. Il chip 74LS155 è abilitato e disabilitato utilizzando l'uscita dal circuito wired-or: la disabilitazione corrisponde a 0 logico (nessun blocco selezionato) e l'abilitazione corrisponde a 1 logico (uno e solo uno dei blocchi di memoria selezionati). La tabella della verità per il chip 74LS155 è data alla pag. 60. Le uscite verso i blocchi 2 e 3 vanno verso gli ingressi $\overline{CE}_1$ (pin 16) della coppia di chip 8111-2 come si può vedere per il blocco 3 nella Fig. 2-9.

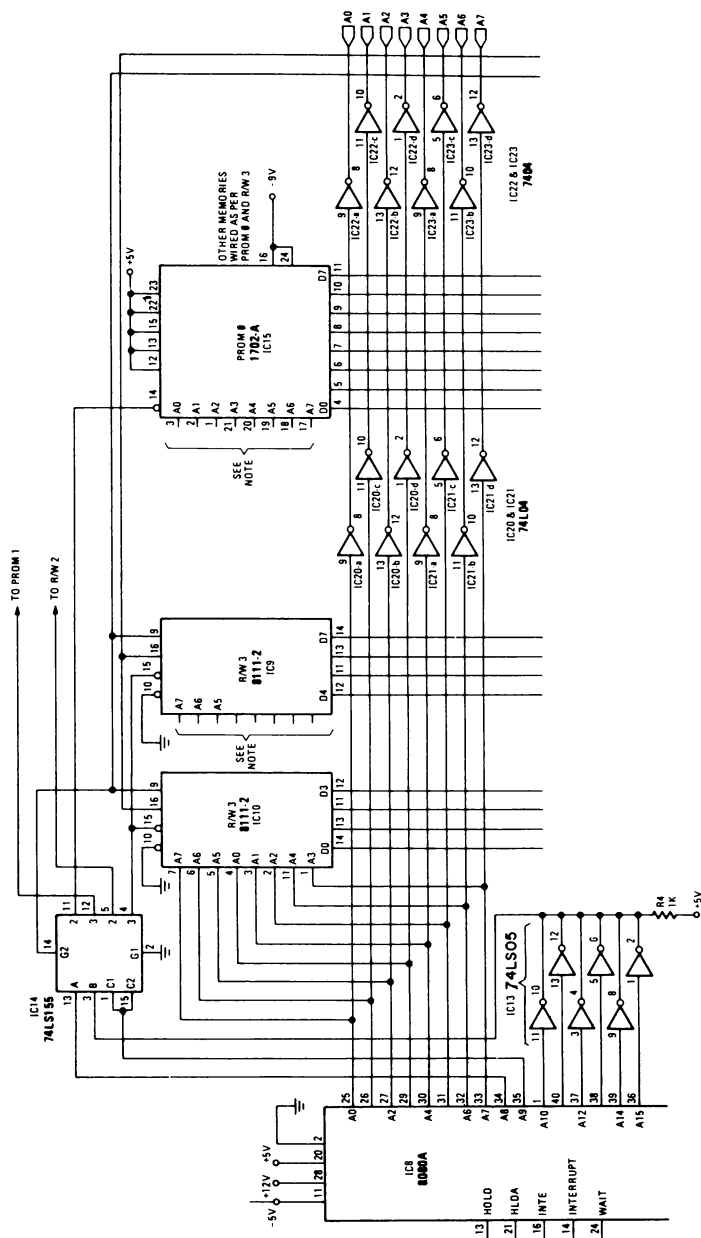



Fig. 2-9. Il bus degli indirizzi, il decoder degli indirizzi e la sezione di memoria del piccolo sistema 8080. Per semplicità, le linee di indirizzi IC9 e IC15 sono state omesse; sono in parallelo con quelle di IC10.

(A) 74LS155 chip. (B) 74155 chip.

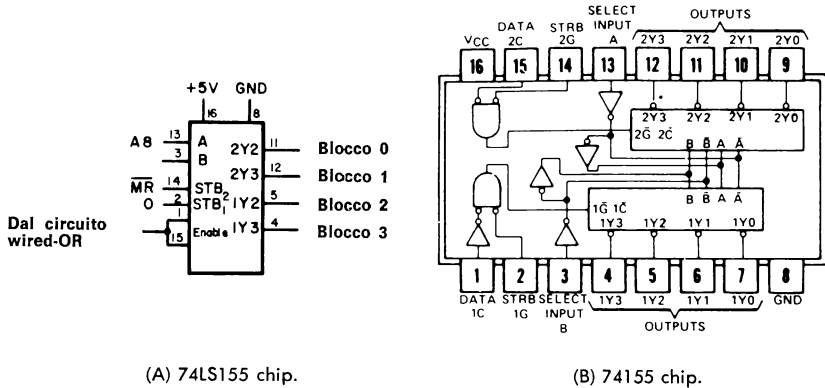



Fig. 2-10. I chip 74155 e 74LS155.

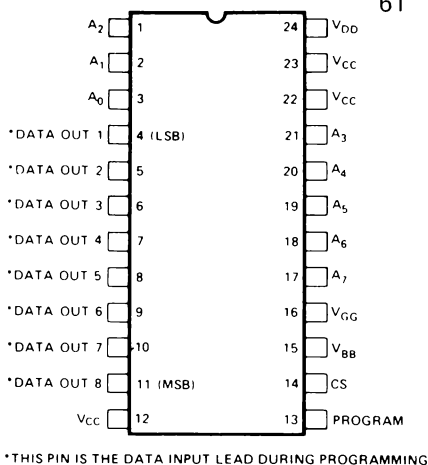
Enable	$\overline{MR}$	*B*	*A*	*2Y2*	*2Y3*	*1Y2*	*1Y3*	*Selezione Blocco*
0	0	X	X	1	1	1	1	Non selezionato
1	0	0	0	0	1	1	1	Blocco 0 (read EPROM memory)
1	0	0	1	1	0	1	1	Blocco 1 (read EPROM memory)
1	0	1	0	1	1	0	1	Blocco 2 (read R/W memory)
1	0	1	1	1	1	1	0	Blocco 3 (read R/W memory)
1	1	0	0	1	1	1	1	Non selezionato
1	1	0	1	1	1	1	1	Non selezionato
1	1	1	0	0	1	1	1	Blocco 2 (write nella R/W memory)
1	1	1	1	1	0	1	1	Blocco 3 (write nella R/W memory)

Quando il pin 15 del chip 8111-2 è a 0 logico, il chip è abilitato con $\overline{CE}_2$ (pin 10) a 0 logico.

Oltre alla memoria lettura/scrittura, il nostro piccolo microcomputer contiene anche della memoria a sola lettura. Le ragioni per avere tale memoria saranno discusse più tardi. Basti per ora che la memoria non è distrutta (cioè i suoi contenuti) al cadere dell'alimentazione, come accade nel caso della memoria R/W. Diciamo allora che la memoria read-only è *non volatile.* Il tipo di memoria a sola lettura che usiamo è un tipo speciale, detta *memoria a sola lettura elettricamente programmabile.* Tale memoria, detta più brevemente EPROM, è ampiamente usata nei prototipi. E' possibile acquistare uno speciale dispositivo elettronico detto EPROM programmer, e programmare le EPROM per vostre speciali applicazioni piuttosto che far fare questo lavoro dai costruttori.

Segnale	Pin	Pin	Segnale
A_2	1	24	V_{DD}
A_1	2	23	V_{CC}
A_0	3	22	V_{CC}
*DATA OUT 1	4 (LSB)	21	A_3
*DATA OUT 2	5	20	A_4
*DATA OUT 3	6	19	A_5
*DATA OUT 4	7	18	A_6
*DATA OUT 5	8	17	A_7
*DATA OUT 6	9	16	V_{GG}
*DATA OUT 7	10	15	V_{BB}
*DATA OUT 8	11 (MSB)	14	CS
V_{CC}	12	13	PROGRAM

*THIS PIN IS THE DATA INPUT LEAD DURING PROGRAMMING

Fig. 2-11. La configurazione dei pin del chip 1702A/8702A. Alcuni dei pin di alimentazione in ingresso sono usati solo durante la programmazione. Noi ipotizzeremo che il chip è stato correttamente programmato prima del suo inserimento nel sistema microcomputer 8080.

Noi usiamo le EPROM 1702A (o 8702 A) della Intel, che possono essere cancellate tramite una sorgente di raggi ultravioletti e riprogrammate per circa cento volte. La configurazione dei pin della 1702A/8702A è mostrata nella Fig. 2-11. Osservate che vi sono 8 linee di indirizzo A0 ÷ A7, ed 8 pin di uscita dei dati, DATA OUT 1 ÷ DATA OUT 8, un input chip select (pin 14) ed alcuni pin di ingressi di alimentazione. La Fig. 2-9 mostra questo chip incorporato nel sistema microcomputer 8080. I pin 12, 13, 15, 22 e 23 sono tutti collegati al + 5 V. I pin 16 e 24 sono connessi al —9 V. L'uscita del blocco 0 del decoder 74LS155 è connesso all'ingresso $\overline{CS}$ del 1702A (pin 14). Osservate che potete solo leggere il chip 1702, infatti è una memoria a sola lettura, non una memoria R/W.

Bus del Microcomputer

Il bus del microcomputer per il nostro piccolo sistema 8080, consiste nei seguenti singoli bus: bus degli indirizzi, bus dei dati e bus di controllo. Il bus degli indirizzi è costituito da 16 linee di indirizzo sottoposte a buffer. Nella Fig. 2-9 è mostrata la «bufferizzazione» dei bit di indirizzo da A0 ad A7. Una coppia di invertitori 7404 sono utilizzati per ogni linea di indirizzo.

Il chip 74L04 ha un fan-in di 0,1, oppure 0,16 mA, ed è uno dei più adatti da utilizzarsi con il microprocessore 8080. I chip 8216 (Fig. 2-4) forniscono un buffer sufficiente per le otto linee di bus dati, D0-D7. Le porte NAND 7400 hanno ciascuna un fan-out di dieci, più che sufficiente per ogni segnale del bus di controllo. RESET o INTERRUPT sono ingressi all'8080. L'uscita INTE può non richiedere un buffer.

Ingresso/Uscita (I/O)

La sezione di I/O del microcomputer è mostrata nella Fig. 2-12. Nei seguenti capitoli di questo libro, acquisirete molta familiarità

con la decodifica I/O, l'uso dei latch 7475 e dei buffer three-state 8095. Quindi, qui faremo solo una discussione di prima approssimazione della sezione di I/O.

Fig. 2-12. La sezione di I/O del piccolo sistema microcomputer 8080A.

Per trasferire 8 bit di dati tra l'ACC ed un dispositivo di I/O, viene posto, sul bus degli indirizzi, un codice dispositivo ad 8 bit, ai bit da A0 ed A7. Per selezionare uno fra i possibili 256 dispositivi, è necessario un decoder. Nella Fig. 2-12, questo decoder è dato dal chip 74L42 e da sei 74LS05 (inverter open-collector) presenti alle linee di indirizzo da A3 ad A7. Cinque degli inverter servono come circuiti wired-or, per la decodifica dei bit di indirizzo da A3 ad A7 in uno stato logico unico, quando le cinque linee sono tutte allo 0 logico. Il principio usato è identico a quello uti-

lizzato per i bit di indirizzo da A10 ad A15, nella sezione di memoria vista. La tabella della verità è.

A7	*A6*	*A5*	*A4*	*A3*	*Q*
0	0	0	0	0	1
X	X	X	X	1	0
X	X	X	1	X	0
X	X	1	X	X	0
X	1	X	X	X	0
1	X	X	X	X	0

Nota: X = 0 logico od 1 logico

Il restante inverter open-collector 74LS05 serve per invertire Q a 0 logico quando i cinque bit di indirizzo sono tutti allo 0 logico. Questa condizione è applicata all'ingresso D del chip 74L42, collegato del resto come un «three-line-to-eight-line decoder», con la seguente tabella della verità:

D	*A2*	*A1*	*A0*	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*Selezione del Canale*
1	X	X	X	1	1	1	1	1	1	1	1	Numero del canale selezionato
0	0	0	0	0	1	1	1	1	1	1	1	Canale 0
0	0	0	1	1	0	1	1	1	1	1	1	Canale 1
0	0	1	0	1	1	0	1	1	1	1	1	Canale 2
0	0	1	1	1	1	1	0	1	1	1	1	Canale 3
0	1	0	0	1	1	1	1	0	1	1	1	Canale 4
0	1	0	1	1	1	1	1	1	0	1	1	Canale 5
0	1	1	0	1	1	1	1	1	1	0	1	Canale 6
0	1	1	1	1	1	1	1	1	1	1	0	Canale 7

I canali 0, 1 e 2 sono sottoposti a gate con il segnale di controllo $\overline{\text{OUT}}$, ed usati per lo strobe delle informazioni dal bus dati verso i latch delle porte 0, 1 e 2 rispettivamente. Il canale 0 è «gated» con il segnale $\overline{\text{IN}}$ e realizza l'input di strobe per i dati presenti al buffer 8095, verso l'8080.

Per finire, una coppia di 74148 «three-line-to-eight-line priority encoder» è usata per la codifica della tastiera a 15 tasti: i numeri da 0 a 7, il tasto SEE/STORE (S), GO (G), HI, LO e 3 tasti addizionali non utilizzati (A, B, C).

Funzionamento del Microcomputer

Potete riferirvi agli articoli precedentemente menzionati in *Radio Electronics* per una descrizione operativa di questo piccolo microcomputer.

Brevemente, si può dire questo: potete immettere programmi tramite la tastiera, indagare i contenuti di memoria, eseguire programmi e porre in uscita informazioni verso le 3 porte di uscita.

Per realizzare tutto ciò è necessario un programma, posto in una EPROM 1702A, nel blocco 0, per essere più chiari. Questo chip preprogrammato è detto *K*eyboard *EX*ecutive, o KEX. Quando attuate lo start del microcomputer, per prima cosa premete RESET, ed il microcomputer va alla locazione di memoria 0000000000000000_2 (cioè HI = 000_8 e LO = 000_8).

A tale locazione il chip 8080A trova la prima istruzione che deve eseguire. Da questo punto in avanti, vi è una serie di istruzioni che realizzano un *bootstrap program*, che permette di operare con il microcomputer.

In funzione dell'uso che fate del microcomputer, potete scrivere programmi di bootstrap per l'input dati da tastiera ASCII, teletype, terminale video, cassetta magnetica e poi porre le informazioni in memoria di lettura/scrittura.

Questo programma può anche contenere delle subroutine speciali per l'input/output da cassetta, perforatore/lettore di nastro, floppy disk.

E' al di là dello scopo di questo capitolo descrivere nel dettaglio questo software. E' sufficiente sapere che sarete senz'altro in grado di sivluppare tale software quando avrete completato la lettura di questo libro.

Una volta reso operativo il microcomputer basato sull'8080, vorrete interfacciarlo con il mondo esterno. Prima di abbandonare questo capitolo, sarebbe appropriato definire cosa è l'interfacciamento ed esporre in modo sommario come sia possibile utilizzare un microcomputer per controllare le operazioni di altri circuiti digitali, ad esempio quello della serie 7400. Dato che qualsiasi macchina o apparecchiatura contiene, nella sua parte elettronica, questo chip, capire come interfacciarsi con la serie 7400, vuol dire essere in grado di affrontare problemi ben più grandi.

COSA SI INTENDE PER INTERFACCIAMENTO?

L'*interfacciamento* può essere definito come l'insieme delle operazioni tese a collegare entità diverse in modo da permettere un loro funzionamento compatibile e coordinato.[1] Per compatibile e coordinato intendiamo anche sincronizzato. Ripetiamo alcune importanti definizioni:

(Sync)	Abbreviazione per sincrono, sincronizzare, ecc.[4]
Synchronization pulse (Impulso di sincronizzazione)	Impulso originato da sistemi di trasmissione e posti in un sistema ricevente allo scopo di tenere al passo l'operatività dei due dispositivi.[4]
Synchronize (Sincronizzare)	Mettere un certo elemento, al passo con un altro.[4]

Synchronous (Sincrono)	Al passo o in fase, applicabile a due dispositivi o macchine. Termine applicabile ad un computer, in cui il realizzarsi di una sequenza di operazioni è controllata da segnali o impulsi di clock.[4]
Synchronous computer (Computer sincrono)	Un computer digitale in cui tutte le operazioni ordinarie sono controllate da segnali prevenuti da un clock master.[4]
Synchronous inputs (Ingressi sincroni)	Quegli ingressi di un flip-flop che non controllano direttamente l'uscita, come nel caso delle porte logiche (gate), ma solo quando è permesso dal clock[4]
Synchronous logic (Logica sincrona)	Il tipo di logica digitale usata in un sistema in cui le operazioni logiche si realizzano in sincronismo con impulsi di clock.
Synchronous operation (Operazione sincrona)	Funzionamento di un sistema sotto il controllo di impulsi di clock.[4]

Possiamo ora definire *computer interfacing* così:

Computer interfacing	La sincronizzazione della trasmissione di dati digitali tra un computer e uno o più dispositivi esterni di imput/output.

Per quanto i dettagli dell'interfacciamento varino a seconda del computer usato, i principi generali sono sempre validi.

Ecco alcune delle caratteristiche comuni:

- I dati digitali che sono trasmessi tra un computer ed un dispositivo di I/O sono sia dei segnali di clock individuali, che intere parole di dati.
- Il computer ed il dispositivo di I/O sono entrambi dispositivi sottoposti a clock od a strobe.
- Il computer invia impulsi di sincronizzazione, detti *device select pulses*, al dispositivo di I/O. Questi segnali sincronizzano e selezionano nello stesso istante di tempo.
- I device select pulses (impulsi di selezione dispositivo) sono generati dal software del computer e dall'hardware costituente l'interfaccia.
- Gli impulsi di selezione dispositivo sono segnali brevi. Per il microprocessore 8080, a 2 MHz, durano 500 ns.
- Singoli impulsi di selezione dispositivo possono essere inviati a singoli dispositivi di ingresso o uscita. In questo caso si parla di *external device addressing (indirizzamento di un dispositivo esterno)*.

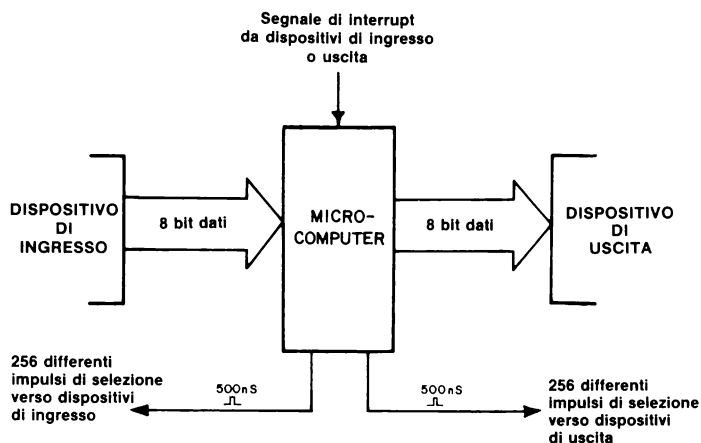



Fig. 2-13. I quattro punti fondamentali dell'interfacciamento: 1) indirizzamento di dispositivi esterni, 2) latch di dati in uscita, 3) strobe di dati in ingresso e 4) servizio delle interruzioni.

- L'indirizzamento di un dispositivo esterno è software generato e decodificato esternamente.
- L'esecuzione di un programma può essere interrotto dalla trasmissione di un impulso di clock da un dispositivo di I/O verso una speciale linea di ingresso al microcomputer.
- A seguito dell'informazione, il microcomputer entra in una speciale subroutine che risponde, o *serve*, all'interruzione.
- Intere parole di dati possono essere poste in uscita dall'accumulatore, od in esse immesse. Per il microcomputer 8080 una intera parola dati contiene 8 bit.
- Tutte le operazioni di trasmissione dati sono sincronizzate dal clock interno al microcomputer.
- I dati in uscita dall'accumulatore sono disponibili solo per un breve periodo di tempo, per cui di solito devono essere sottoposti a *latch.*
- I dati in ingresso verso l'accumulatore possono essere acquisiti in un tempo molto breve, per cui è necessario che siano sottoposti a *strobe* verso l'accumulatore.
- Gli impulsi di selezione dispositivo sono usati per il latch dei dati in uscita e per lo strobe dei dati in ingresso.

Come si può vedere dalla Fig. 2-3 e dai precedenti commenti, *i 4 compiti principali dell'interfacciamento sono:*

- Indirizzamento di dispositivi esterni tramite la generazione di impulsi di selezione dispositivo.
- Il latch di dati in uscita.
- Lo strobe di dati in ingresso.
- Il servizio delle interruzioni.

COSA SI INTENDE PER DISPOSITIVO DI I/O?

Alcune utili definizioni sono:

Input/Output (Ingresso/ Uscita)	Termine generale per tutto ciò che serve per comunicare con un computer e per gestire i dati coinvolti nella comunicazione.[5]
I/O	Abbreviazione di input/output.
I/O device (Dispositivo di I/O)	Qualunque dispositivo digitale, incluso il singolo circuito integrato, che trasmette dati o riceve dati o realizza la strobe di impulsi da un microcomputer.

Da un punto di vista tradizionale, un dispositivo di I/O è un qualcosa di complesso ed ingombrante come giustamente sono i lettori di schede, le unità a nastro magnetico, i display, la teletype.

Però ora, *un singolo circuito integrato, come un latch, shift register counter, possono essere senz'altro considerati come dispositivi di I/O per un microcomputer.*

Un'altra importante considerazione, è che spesso più impulsi di selezione dispositivo sono necessari per un singolo dispositivo di I/O. Ad esempio, lo shift register 74198 possiede una coppia di ingressi di controllo che determinano se lo shift deve avvenire a destra, a sinistra oppure se si devono, in parallelo, caricare 8 bit di dati. C'è poi un input di clock e di clear. Così il chip 74198 può richiedere fino a 4 linee di device select pulse da un microcomputer. Quindi, la possibilità di generare 256 impulsi di selezione dispositivo sia in ingresso che in uscita, non significa necessariamente che si possono indirizzare 512 «differenti» dispositivi: più ragionevolmente, si può parlare di 50 o 100 al massimo.

E' facile, infine, realizzare gli impulsi di selezione dispositivo. Dovreste utilizzarli frequentemente, *col fine di sostituire col software, la logica cablata.*

Ricordatevi questo obiettivo: software al posto dell'hardware. Facendo ciò, la sola penalità che pagate è il tempo, nel senso che occorre tempo per eseguire le istruzioni di un dato computer. Se potete accettare questo carico, potete allora ampiamente semplificare la circuiteria richiesta per realizzare uno specifico compito.

UTILIZZO DEGLI IMPULSI DI SELEZIONE DISPOSITIVO

Abbiamo precedentemente definito un clock sia come (a) qualsiasi dispositivo che genera un impulso di clock che come (b) un dispositivo di temporizzazione in un sistema che fornisce una serie continua di impulsi di clock.

Un *multivibratore monostabile* è un circuito con un solo stato stabile, da cui può essere, attraverso trigger, spostato, ma solo per brevi intervalli di tempo, dopo cui ritorna allo stato di partenza. Un *pulser* è uno switch logico che genera un singolo impulso di clock.

Con queste 3 definizioni in mente, possiamo dire che un microcomputer è, tra l'altro, *un circuito elettronico sofisticato che può agire come clock,* come pulser, *come multivibratore monostabile.*

Ad esempio, l'8080 a 2 MHz può:

- Generare singoli impulsi di clock da 500 ns, in qualunque istante.
- Generare un treno di impulsi di clock, la cui frequenza è data dalla formula $f = 2/n$ MHz, dove n è un intero qualunque maggiore o eguale a 20.
- Generare singoli impulsi, la cui ampiezza è data dalla formula $\tau = n/2$ µs dove n è un intero qualunque maggiore od eguale a 10; è però necessario, per questo, un flip-flop esterno.

Il microcomputer realizza quanto detto con l'aiuto degli impulsi di selezione dispositivo la cui frequenza, ma non l'ampiezza, è determinata dal software del microcomputer. Gli impulsi, si dice siano *software generated* (generati dal software) cioè nascono nell'ambito del programma del microcomputer. Quindi:

- Se un impulso di clock isolato è generato dal programma, allora il microcomputer agisce come un pulser oppure come un dispositivo di strobe.
- Se il programma contiene un *loop di timing,* come è una serie di impulsi di clock, generati ad intervalli di tempo ripetitivi, allora il microcomputer agisce come un clock.
- Se una coppia di impulsi di clock è generata per il preset ed il clear di un flip-flop, allora la combinazione microcomputer/flip-flop agisce da multivibratore monostabile.

Così facendo, *si è sostituito l'hardware con il software,* eliminando la necessità di pulser, clock o multivibratori.

Il microcomputer basato sul microprocessore 8080 può generare 256 differenti impulsi di clock in uscita e 256 in ingresso, di 500 ns, con opportuni circuiti di decodifica. E' cioè possibile avere:

512 differenti pulser o dispositivi elettronici di strobe o
512 differenti clock
512 differenti multivibratori monostabili

o qualsiasi combinazione di essi.

UTILIZZO DI UN MICROCOMPUTER PER REALIZZZARE LO STROBE DI CIRCUITI INTEGRATI

Forse, l'applicazione più importante per un microcomputer è abilitare, guidare, l'operatività di strumenti, dispositivi elettronici, circuiti integrati. I chip di circuito integrato sono a basso costo, e con cui è possibile dimostrare l'intero range delle applicazioni degli impulsi di selezione dispositivo generati da un microcomputer. Tali impulsi possono, ad esempio:

- Azzerare contatori, shift register, flip-flop e latch.
- Caricare contatori, latch e shift register.
- Abilitare multiplexer, demultiplexer, decoder, contatori, latch, shift register, memorie, priority encoder, ed altri chip.
- Inibire ingressi di clock verso contatori e shift register.
- Settare, azzerare, fornire clock a flip-flop.
- Selezionare la direzione di shift ed iniziare eventuali funzioni negli shift register.

Tutto quanto elencato nei punti precedenti realizza *la sostituzione dell'hardware con il software.*

Nostro obiettivo fondamentale con i microcomputer ed i microprocessori è sostituire l'hardware con il software!

Questo testo ha un duplice scopo: (a) insegnarvi come avviene l'interfacciamento con un microcomputer, e (b) mostrarvi come sostituire l'hardware con il software.

Per enfatizzare il precedente messaggio, vi proponiamo un sunto delle caratteristiche di alcuni dei più noti circuiti integrati della serie 7400, dal punto di vista dello «strobe».

I chip sono raggruppati per funzione ed è identificato il numero del pin che interessa.

Per ogni pin, sarà indicato lo stato logico richiesto per realizzare la funzione di strobe.

Contatori

Tra i più noti contatori vi sono il 7490, 7493, 74192 e 74193. Altri meno popolari sono: 74160, 74161, 74163, 74191.

7490 pin 2 e 3: 1 logico ad entrambi i pin realizza il clear del contatore.
pin 6 e 7: 1 logico ad entrambi i pin setta il contatore a 9.
pin 14: input di clock.

7493 pin 2 e 3: 1 logico ad entrambi i pin realizza il clear del counter.
pin 14: input di clock.

74163 pin 1: 0 logico azzera il counter

da 74160	pin 2: input di clock. pin 7: 0 logico inibisce il counter. pin 9: 0 logico abilita il flip-flop e permette il caricamento del counter. pin 10: 0 logico inibisce il counter ed il ripple carry.
a 74190	pin 4: 1 logico inibisce il counter.
74191	pin 5: 0 logico per Up counter ed 1 logico per Down counter. pin 11: 0 logico abilita i flip-flop e permette il caricamento del counter. pin 14: clock input.
74192	pin 4: input di clock per il Down counter; 1 logico disabilita.
74193	pin 5: input di clock per l'Up counter, 1 logico disabilita. pin 11: 0 logico abilita i flip-flop e permette al counter di essere caricato.

Decoder

Vi sono degli ingressi di «enable» (abilitazione) per il 74LS1389, 74LS139, 74154 e 74155. I decoder BCD-decimale, possono essere convertitori in «strobed octal decoder», se l'ingresso D è usato come ingresso di strobe.

7442, 7445	pin 12: 0 logico abilita il decoder ottale.
7446, 7447, 7448	pin 3: lamp test; 0 logico accende tutti e sette i segmenti.
74LS138	pin 4, 5, 6: 0 logico a 4 e 5 ed 1 a 6, abilitano il decoder.
74LS139	pin 1: 0 logico abilita il primo decoder. pin 15: 0 logico abilita il secondo decoder.
74154	pin 18, 19: 0 logico al 18 e 19, abilita il decoder.
74155	pin 1: 1 logico abilita il primo two-line-to-four-line-decoder. pin 2: 0 logico abilita il primo two-line-to-four-line-decoder. pin 13: 1 logico abilita il secondo two-line-to-four-line-decoder. pin 14: 0 logico abilita il secondo two-line-to-four-line-decoder. pin 12 e 14 (connessi assieme): 0 logico abilita il three-line-to-eight-line decoder.

Demultiplexer (demux)

Un decoder può essere collegato come demux. I 74LS138, 74154 e 74155 decoder/demux hanno ciascuno degli input di enable.

74LS138	pin 4, 5: 0 logico a 4 e 5 abilita il demux.
74154	pin 19: 0 logico abilita il demux.
74155	pin 2: 0 logico abilita il primo demux. pin 14: 0 logico abilita il secondo demux.

Data Selector Multiplexer (mux)

Vi sono ingressi enable per ciascuno dei tre data selector/mux della serie 7400, cioè il 74150, 74151 e 74153, come pure per i mux 74156, 74157 e 74158.

74150	pin 9: 0 logico abilita il data selector/mux.
74151	pin 7: 0 logico abilita il data selector/mux.
74153	pin 1: 0 logico abilita il primo data selector/mux. pin 15: 0 logico abilita il secondo data selector/mux.
74156-58	pin 1: 0 logico seleziona gli input A, 1 logico seleziona gli input B. pin 15: 0 logico abilita il data selector/mux.

Shift Register

Descriveremo solo i più recenti shift register della serie 7400, e cioè: 74164, 74165, 74166, 74194, 74198 e 74199.

74164	pin 8: 0 logico azzera il registro. pin 9: input di clock.
74165	pin 1: 0 logico carica il registro; 1 logico realizza lo shift dei dati. pin 2: input di clock. pin 15: 1 logico inibisce il clock.
74166	pin 6: 1 logico inibisce il clock. pin 7: input di clock. pin 9: 0 logico azzera il registro. pin 15: 0 logico carica il registro; 1 logico realizza lo shift dei dati.
74194	pin 1: 0 logico azzera il registro. pin 9 e 10: mode select inputs; il pin 9 è S0 ed il 10 è S1; se S0 = 0 e S1 = 0, il clock è inibito. S0 = 1 e S1 = 0, shift right; S0 = 0 e S1 = 1, shift left; S0 = 1 e S1 = 1, caricamento parallelo. pin 11: clock input.
74198	pin 1 e 23: mode select inputs; pin1 è S0 e pin 23 è S1; S0 = 0 e S1 = 0, il clock è inibito; S0 = 1 e S1 = 0, shift right; S0 = 0 e S1 = 1, shift left; S0 = 1 e S1 = 1, caricamento parallelo.

	pin 13: 0 logico azzera il registro.
	pin 11: clock input.
74199	pin 11: 1 logico inibisce il clock.
	pin 13: input di clock.
	pin 14: 0 logico azzera il registro.
	pin 23: 0 logico realizza il caricamento in parallelo del registro; 1 logico realizza lo shift dei dati a destra.

Priority Encoder

Parleremo solo del 74148, eight-data-line to three-line.

74148	pin 5: 0 logico abilita il chip.

Latch e Flip-flop

I chip di latch contengono 4 o più flip-flop di tipo D. I più comuni latch della serie 7400 includono: 7475, 74100, 74116, 74173, 74174 e 74175.

7475	pin 3: 1 logico abilita i primi due latch.
	pin 13: 1 logico abilita i secondi due latch.
74100	pin 23: 1 logico abilita i primi 4 latch.
	pin 12: 1 logico abilita i secondi 4 latch.
74116	pin 1: 0 logico azzera i primi 4 latch.
	pin 2 e 3: 0 logico a 2 e 3 abilita i primi 4 latch.
	pin 13: 0 logico azzera i secondi 4 latch.
	pin 14 e 15: 0 logico a 13 e 15 abilita i secondi 4 latch.
74173	pin 1 e 2: 0 logico a 1 e 2 abilita le uscite three-state.
	pin 7: input di clock.
	pin 9 e 10: 0 logico a 9 e 10 abilita i registri.
	pin 15: 1 logico azzera i registri.
74174	pin 9: input di clock.
74175	pin 1: 0 logico azzera i latch.

I flip-flop della serie 7400, come 7470, 7473, 7474, 7476, 74106 ecc., hanno troppi differenti ingressi per essere elencati. I tipi di ingressi sono: preset, clear, clock, R, S, J, K. Vediamo alcuni esempi:

7470	pin 2: 0 logico azzera il flip-flop.
	pin 3, 4, 5: ingressi J.
	pin 9, 10, 11: ingressi K.
	pin 12: ingressi di clock.
	pin 13: 0 logico setta il flip-flop.
7473	pin 1: input di clock al primo flip-flop.
	pin 2: 0 logico azzera il primo flip-flop.
	pin 3: ingresso K al primo flip-flop.
	pin 14: ingresso J al primo flip-flop.

pin 5: 0 logico abilita il chip.
pin 6: 0 logico azzera il secondo flip-flop.
pin 7: ingresso J al secondo flip-flop.
pin 10: ingresso K al secondo flip-flop.

7474 pin 1: 0 logico azzera il primo flip-flop.
pin 2: ingresso D al primo flip-flop.
pin 3: ingresso di clock al primo flip-flop.
pin 4: 0 logico setta il primo flip-flop.
pin 10: 0 logico setta il secondo flip-flop.
pin 11: ingresso di clock al secondo flip-flop.
pin 12: ingresso D al secondo flip-flop.
pin 13: 0 logico azzera il secondo flip-flop.

Memorie

Diamo, di seguito, le caratteristiche di una certa varietà di memorie.

7488 pin 15: 0 logico abilita la memoria.
7489, pin 2: 0 logico abilita la memoria.
8225 pin 3: 0 logico scrittura; 1 logico lettura.
74200, pin 3, 4 e 5: 0 logico a 3, 4 e 5 abilitano la memoria.
74206 pin 12: 0 logico scrittura, 1 logico lettura.
2102, pin 3: 0 logico scrittura, 1 logico lettura.
8102 pin 13: 0 logico abilita la memoria.
1602A, pin 14: 0 logico abilita la memoria.
1702A,
8702A,
1302,
8302

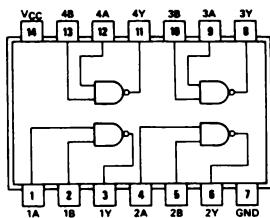
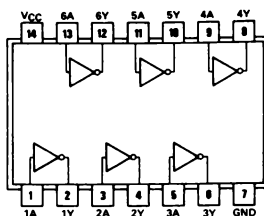
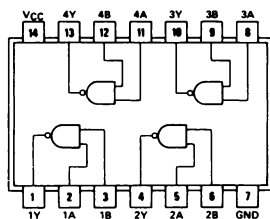
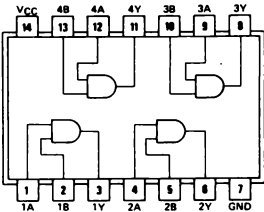
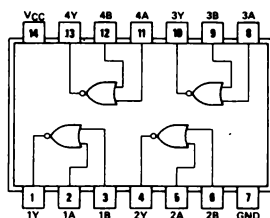
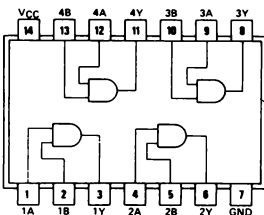
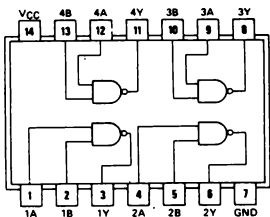
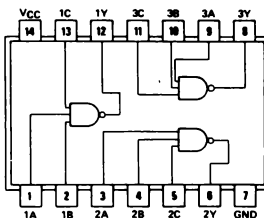
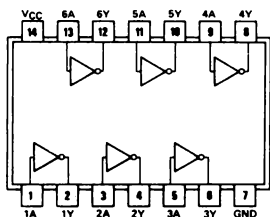
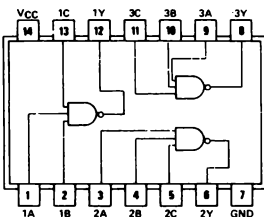
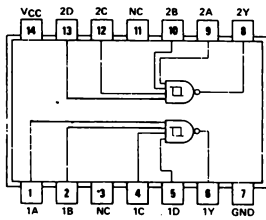
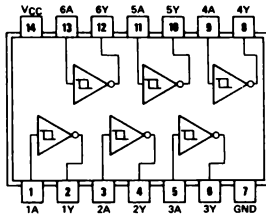
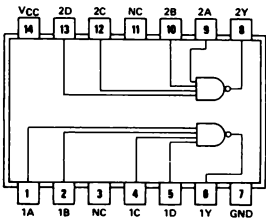
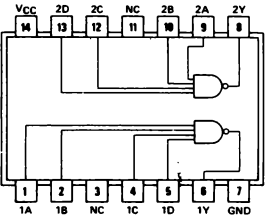
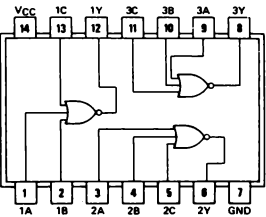
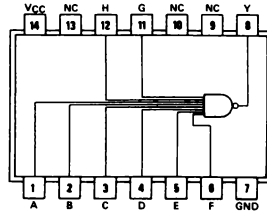
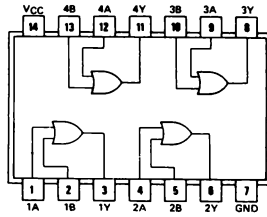
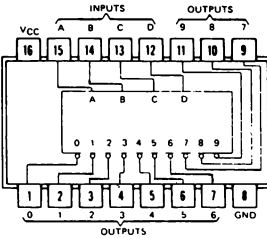
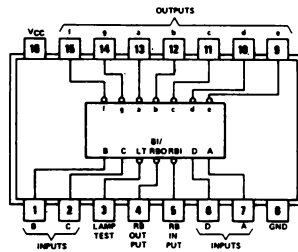
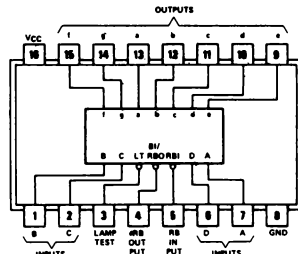
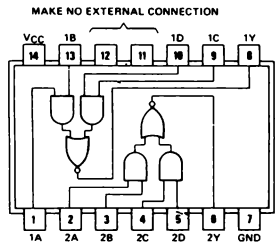
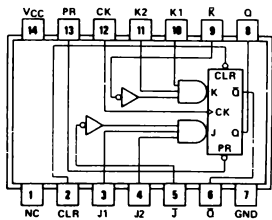
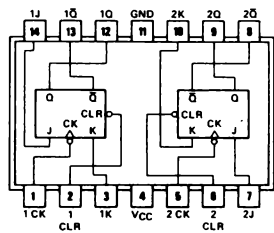
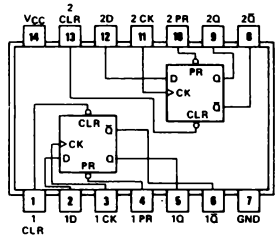
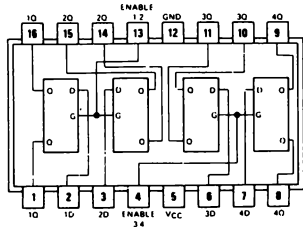
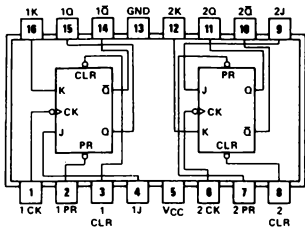
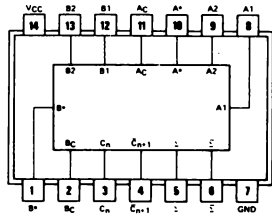
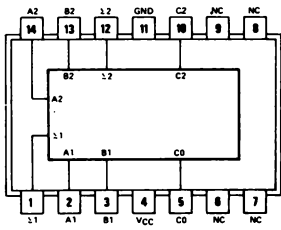
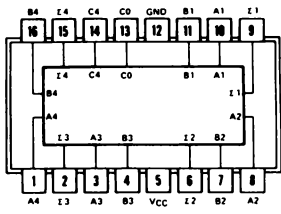
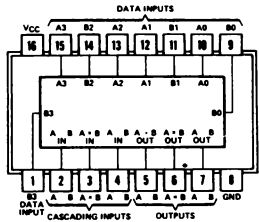
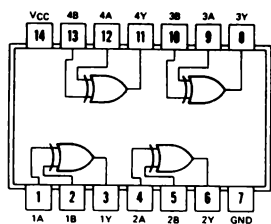
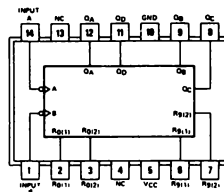
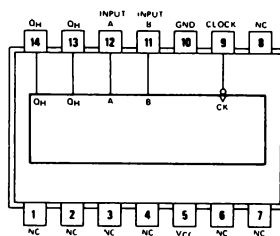
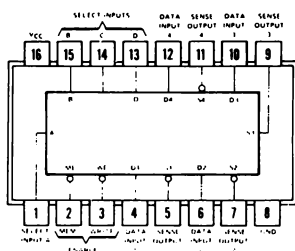
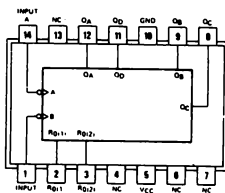
Altri Chip

Diamo di seguito alcuni chip three-state, di supporto al microprocessore 8080.

8212 pin 1 e 13: entrambi all'1 logico, abilitano il chip
pin 2: 1 logico latch dei dati ed abilitazione delle uscite three-state.
pin 11: 1 logico
pin 14: 0 logico azzera i latch.

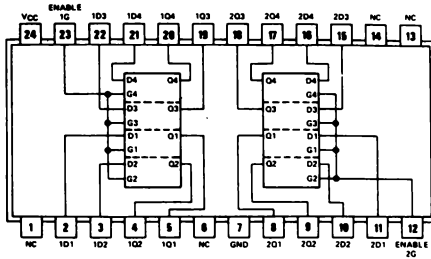
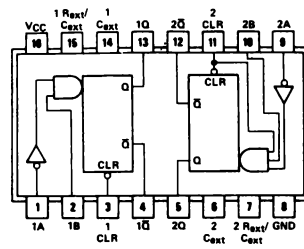
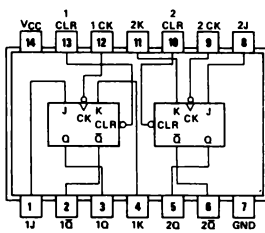
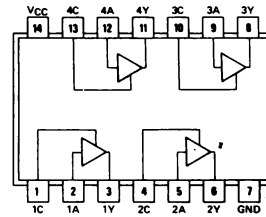
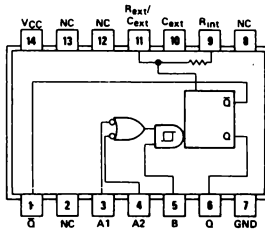
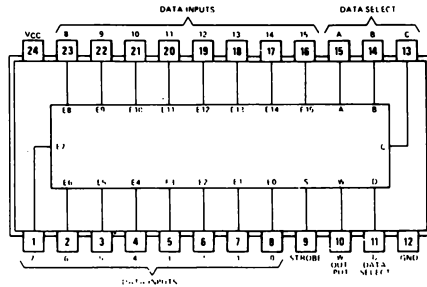
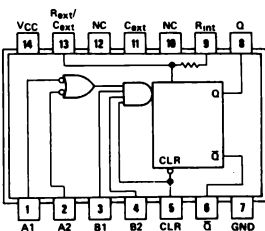
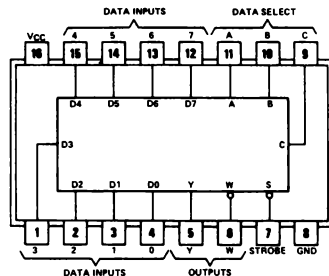
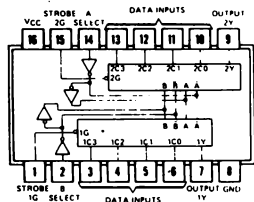
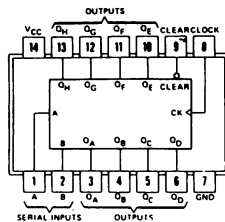
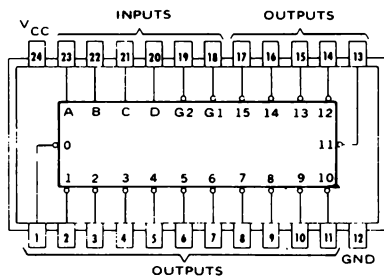
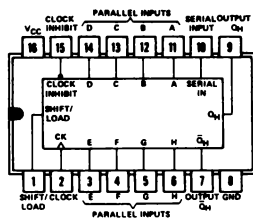
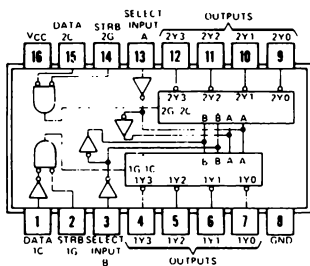
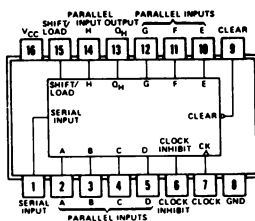
8255 pin 5: 0 logico lettura dati.
pin 6: 0 logico abilita il chip.
pin 36: 0 logico scrittura dati.

Configurazioni dei pin

Le configurazioni dei pin di 72 differenti circuiti integrati, sono date nelle seguenti pagine, su gentile concessione della Texas Instruments, Inc. e della Intel Corp.

Courtesy Texas Instruments, Inc.

Courtesy Texas Instruments, Inc.

Courtesy Texas Instruments, Inc.

7486

7490

7491

7493

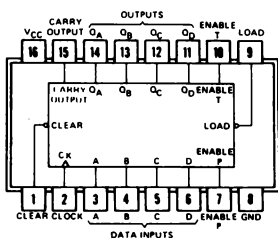
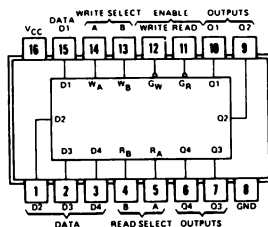
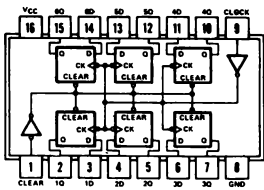
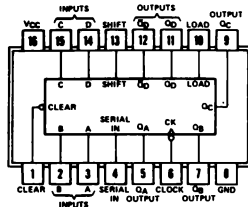
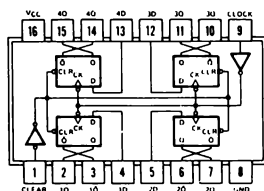
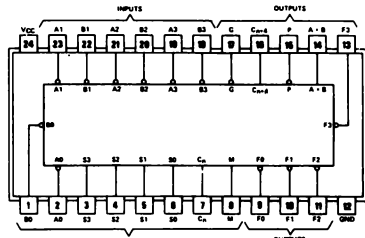
RESULTANT DISPLAYS USING '46A, '47A, '48, '49, 'L46, 'L47

Courtesy Texas Instruments, Inc.

Courtesy Texas Instruments, Inc.

74153

74164

74154

74165

74155

74166

74160 to 74163

74170

Courtesy Texas Instruments, Inc.

74174

74179

74175

74181

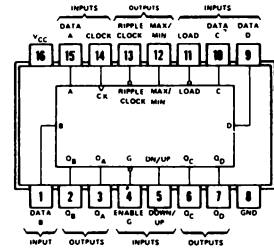
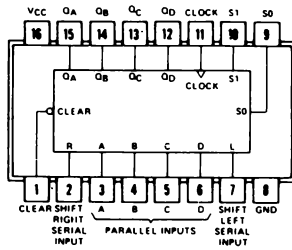
74181

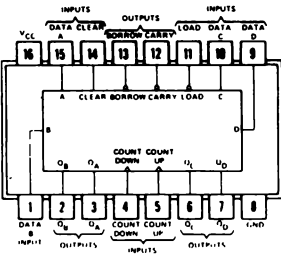
TABLE 1

SELECTION S3 S2 S1 S0	ACTIVE HIGH DATA: M = H LOGIC FUNCTIONS	M = L ARITHMETIC OPERATIONS: C_n = H (no carry)	M = L ARITHMETIC OPERATIONS: C_n = L (with carry)
L L L L	F = $\overline{A}$	F = A	F = A PLUS 1
L L L H	F = $\overline{A + B}$	F = A + B	F = (A + B) PLUS 1
L L H L	F = $\overline{A}$B	F = A + $\overline{B}$	F = (A + $\overline{B}$) PLUS 1
L L H H	F = 0	F = MINUS 1 (2's COMPL)	F = ZERO
L H L L	F = $\overline{AB}$	F = A PLUS A$\overline{B}$	F = A PLUS A$\overline{B}$ PLUS 1
L H L H	F = $\overline{B}$	F = (A + B) PLUS A$\overline{B}$	F = (A + B) PLUS A$\overline{B}$ PLUS 1
L H H L	F = A $\odot$ B	F = A MINUS B MINUS 1	F = A MINUS B
L H H H	F = A$\overline{B}$	F = A$\overline{B}$ MINUS 1	F = A$\overline{B}$
H L L L	F = $\overline{A}$ + B	F = A PLUS AB	F = A PLUS AB PLUS 1
H L L H	F = $\overline{A \odot B}$	F = A PLUS B	F = A PLUS B PLUS 1
H L H L	F = B	F = (A + $\overline{B}$) PLUS AB	F = (A + $\overline{B}$) PLUS AB PLUS 1
H L H H	F = AB	F = AB MINUS 1	F = AB
H H L L	F = 1	F = A PLUS A*	F = A PLUS A PLUS 1
H H L H	F = A + $\overline{B}$	F = (A + B) PLUS A	F = (A + B) PLUS A PLUS 1
H H H L	F = A + B	F = (A + $\overline{B}$) PLUS A	F = (A + $\overline{B}$) PLUS A PLUS 1
H H H H	F = A	F = A MINUS 1	F = A

*Each bit is shifted to the next more significant position.

TABLE 2

SELECTION S3 S2 S1 S0	ACTIVE LOW DATA: M = H LOGIC FUNCTIONS	M = L, ARITHMETIC OPERATIONS: C_n = L (no carry)	M = L, ARITHMETIC OPERATIONS: C_n = H (with carry)
L L L L	F = $\overline{A}$	F = A MINUS 1	F = A
L L L H	F = $\overline{AB}$	F = AB MINUS 1	F = AB
L L H L	F = $\overline{A}$ + B	F = A$\overline{B}$ MINUS 1	F = A$\overline{B}$
L L H H	F = 1	F = MINUS 1 (2's COMP)	F = ZERO
L H L L	F = $\overline{A + B}$	F = A PLUS (A + $\overline{B}$)	F = A PLUS (A + $\overline{B}$) PLUS 1
L H L H	F = $\overline{B}$	F = AB PLUS (A + $\overline{B}$)	F = AB PLUS (A + $\overline{B}$) PLUS 1
L H H L	F = $\overline{A \odot B}$	F = A MINUS B MINUS 1	F = A MINUS B
L H H H	F = A + $\overline{B}$	F = A + $\overline{B}$	F = (A + $\overline{B}$) PLUS 1
H L L L	F = $\overline{A}$B	F = A PLUS (A + B)	F = A PLUS (A + B) PLUS 1
H L L H	F = A $\odot$ B	F = A PLUS B	F = A PLUS B PLUS 1
H L H L	F = B	F = A$\overline{B}$ PLUS (A + B)	F = A$\overline{B}$ PLUS (A + B) PLUS 1
H L H H	F = A + B	F = A + B	F = (A + B) PLUS 1
H H L L	F = 0	F = A PLUS A*	F = A PLUS A PLUS 1
H H L H	F = A$\overline{B}$	F = AB PLUS A	F = AB PLUS A PLUS 1
H H H L	F = AB	F = A$\overline{B}$ PLUS A	F = A$\overline{B}$ PLUS A PLUS 1
H H H H	F = A	F = A	F = A PLUS 1

Courtesy Texas Instruments, Inc.

Courtesy Texas Instruments, Inc.

2102

*THIS PIN IS THE DATA INPUT LEAD DURING PROGRAMMING.

1702A

8080

Courtesy Intel Corp.

Courtesy Intel Corp.

COSA AVETE REALIZZATO IN QUESTO CAPITOLO?

Nella prima parte di questo capitolo era stato detto che alla fine, sareste stati in grado di:

- Identificare il bus degli indirizzi e dei dati, degli ingressi e delle uscite di controllo, degli ingressi di alimentazione, con riferimento al microprocessore 8080A.

  Una descrizione del microprocessore 8080A è stata data all'inizio del capitolo. Benché possiate non avere del tutto capito la funzione di alcuni pin, certamente sarete ora capaci di identificarli tutti.
- Descrivere le funzioni di ciascun pin sul microprocessore 8080A.

  La descrizione di ciascun pin è stata infatti data. Una volta letto il Capitolo 6, avrete la possibilità di capire meglio gli I/O di controllo dell'8080.
- Descrivere in dettaglio le varie parti costituenti un microcomputer basato sull'8080.

  Il microcomputer mostrato nelle Figg. 2-4, 2-12 è stato senz'altro dettagliatamente descritto. Dovreste essere capaci di spiegare le funzioni dei seguenti chip: 8224, 8216, 74174, 8111-2 e 1702-A.
- Elencare i principi generali dell'interfacciamento, applicabili ai computer digitali.

  Questo è stato fatto nella parte finale del capitolo.
- Spiegare cosa è un dispositivo di I/O.

  Può essere un qualsiasi dispositivo digitale, incluso un circuito integrato, che trasmette dati o che li riceve.
- Elencare tre importanti utilizzi degli impulsi di selezione dispositivo.

  Possono servire come sorgenti di impulsi di clock, oppure per abilitare latch, mux, decoder e shift register. Possono azzerare contatori, registri e settare, azzerare flip-flop.
- Elencare gli ingressi ai circuiti più comuni della serie 7400, che possono essere sottoposti a strobe con gli impulsi di selezione dispositivo generati da un microcomputer.

  Questo è stato fatto in considerevole dettaglio in questo capitolo.

CAPITOLO 3

Una Introduzione alla Programmazione dei Microcomputer

In questo capitolo, imparerete le caratteristiche del set di istruzioni del microprocessore 8080, cioè le 78 istruzioni base più le 244 (di cui 12 non usate) derivate dal set base. Non dovete, in questo capitolo, scrivere dei programmi, ma vi saranno solo sottoposti degli esempi di programmi, da studiare e valutare. E' opinione degli autori che la programmazione dei microcomputer è spesso di più semplice apprendimento, che non le tecniche di interfacciamento. Esistono parecchi testi e manuali sui vari microprocessori e la maggior parte sembrano occuparsi quasi esclusivamente delle caratteristiche della programmazione. Dovreste studiare questi libri per quel tanto che valgono. Noi, che puntiamo all'interfacciamento, ci limiteremo a studiare bene quelle istruzioni che saranno più utili a questo scopo.

OBIETTIVI

Alla fine di questo capitolo sarete in grado di:

- Spiegare quale è la differenza tra un'istruzione, un'operazione, un programma, un'istruzione in codice macchina, un'istruzione in linguaggio assembler, un'istruzione mnemonica.
- Definire i termini: assembler, bit, byte, flag, simbolo mnemonico, codice dispositivo, byte di indirizzo HI, byte di indirizzo LO, incremento, decremento, etichetta, salto (Jump), salto a subroutine (call), ritorno da subroutine, operando, flag di

carry, flag di parità, flag di zero, flag di segno, registro, coppia di registri, subroutine, istruzione a due byte, istruzione a tre byte, operazione incondizionata e condizionata, branch, stack pointer, program counter, accumulatore, ALU, byte di dati, registro istruzione.

- Classificare le istruzioni dell'8080 in 5 gruppi.
- Spiegare come un'istruzione ad 8 bit può essere scritta sia in codice ottale che esadecimale.
- Elencare i codici mnemonici di almeno dieci differenti istruzioni del microprocessore 8080.
- Spiegare la differenza tra linguaggio macchina e linguaggio assembler.
- Elencare i registri che sono presenti nell'8080.
- Spiegare come il microprocessore capisce cosa fare per una data istruzione.
- Spiegare come il microprocessore decodifica:

  Le classi di istruzioni
  I registri
  Le coppie di registri
  Le operazioni di branch
  Le operazioni di incremento
  Le operazioni di decremento

DEFINIZIONI

Accumulator (Accumulatore) — Il registro e l'associata circuiteria elettronica nell'unità aritmetica di un computer, in cui si realizzano operazioni logiche ed aritmetiche.

Address (Indirizzo) — Nel microprocessore 8080, un numero a 16 bit che identifica una locazione di memoria.

ALU — Abbreviazione per Unità Aritmetico-Logica. Sottosistema di calcolo che realizza le operazioni matematiche e logiche di un sistema digitale.[3]

Arithmetic operation (Operazione aritmetica) — Addizione, sottrazione, moltiplicazione, divisione e confronto.

To assemble (Assemblare) — Tradurre da simbolico a binario, sostituendo codici operativi binari ai codici simbolici e sostituendo agli indirizzi simbolici gli indirizzi assoluti e rilocabili.[4]

Assembler (Assemblatore) — Un programma che genera un programma in linguaggio macchina, a partire da un programma scritto in linguaggio simbolico, sostituendo codici operativi binari ai codici simbolici e indirizzi assoluti e rilocabili agli indirizzi simbolici.[4]

Assembly (Assemblaggio)	Processo in cui le istruzioni simboliche sono tradotte in binario da un computer.
Assembly language (Linguaggio assembler)	Un linguaggio che ha una corrispondenza uno ad uno con un programma assembler.[4]
Assembly language programming (Programmazione in linguaggio assembler)	La scrittura di un programma in un linguaggio che facilita la traslazione di programmi in codice binario, tramite l'uso di simboli mnemonici.[4]
Auxiliary carry flag (Flag di carry ausiliario)	Un flip-flop che va allo stato logico 1 quando c'è un carry dal bit 3 verso il bit 4 nel microprocessore 8080 durante operazioni quali la addizione, la sottrazione, il confronto. Usato soprattutto con somme precedenti un'istruzione di Decimal Adjust Accumulator.
Bit	La più piccola unità di informazione che può essere rappresentata. Un bit può essere in uno tra due stati, 0 o 1 logico.
Branch instruction (Istruzione di branch)	Un'istruzione che determina un salto verso una specifica locazione, da cui l'esecuzione dell'istruzione a quell'indirizzo. Durante l'esecuzione di una branch, la CPU sostituisce il contenuto del program counter con lo specifico indirizzo.
Branch operation (Operazione di branch)	Vedi istruzione di branch.
Byte	Un gruppo di 8 bit contigui, occupanti una singola locazione di memoria nel microprocessore 8080.
Call	Uno speciale tipo di salto in cui la CPU deve «ricordare» il contenuto del program counter nel momento in cui il salto si verifica. Questo permette di riprendere l'esecuzione del programma principale, una volta esaurita la subroutine.
Call subroutine	Vedi call.
Carry flag (Flag di carry)	Un flip-flop che si porta nello stato logico 1 quando c'è un carry ad un borrow dal bit di ordine più alto durante un'operazione aritmetica; altrimenti è posto a zero.

Computer instruction (Istruzione di un computer)	Un set di caratteri che definiscono un'operazione, unitamente ad uno o più indirizzi, oppure no, e che, come unità, fa si che il computer attui la data operazione sulle quantità indicate.[5]
Computer program (Programma di un computer)	Una sequenza di istruzioni che, prese come gruppo, permettono al computer di realizzare un desiderato compito.[7]
Conditional (Condizionato)	Un computer, soggetto al risultato di un confronto eseguito durante un calcolo.[4]
Conditional break point instruction (Istruzione condizionata da break point)	Un'istruzione di salto condizionato che determina un arresto del computer, se un dato switch è settato. La routine può poi essere abilitata a procedere come codificato, oppure può essere forzato un salto.[4]
Conditional jump (Salto condizionato)	Detto anche trasferimento di controllo condizionato. Un'istruzione ad un computer che causerà l'utilizzo di uno tra due, o più, indirizzi, al fine di ottenere la successiva istruzione, in funzione delle caratteristiche di una o più espressioni numeriche, od altra condizione.[4]
Condition flag (Flag di condizione)	Vedi flag.
Data byte (Byte di dati)	Un numero binario ad 8 bit che il microprocessore 8080 usa nelle operazioni aritmetiche o logiche, oppure che memorizza nel suo sistema di memoria.
Decrement (Decremento)	La diminuzione del valore di una parola binaria. Tipicamente il decremento si riferisce alla diminuzione di uno, del valore iniziale.
Destination register (Registro destinazione)	Il registro che riceve una parola dati ad 8 bit, sottoposto a trasferimento.
Device code (Codice dispositivo)	Codice ad 8 bit per uno specifico dispositivo di I/O. Questo codice è decodificato da decoder esterni che, con gli impulsi IN o OUT del microprocessore 8080, generano un singolo device select pulse.
Direct addressing (Indirizzamento diretto)	Il byte di dati è acquisito tramite un'istruzione a 3 byte che contiene nel 2° e 3° byte l'indirizzo di memoria a 16 bit in cui è posto il dato acquisito.

Field (Campo)	Un gruppo di bit in un byte od in una parola, trattato come singola unità di informazione. Usualmente il numero di bit nel campo è specificato come, ad esempio, «three-bit field» (campo a 3 bit).[7]
Flag (Flag)	Un singolo flip-flop che indica che si sono verificate certe condizioni durante operazioni logiche od aritmetiche oppure durante la trasmissione dati tra due dispositivi digitali elettronici. Ad esempio, un flag può essere un circuito che fornisce un segnale atto ad indicare che un certo dispositivo di I/O è pronto a ricevere o trasmettere dati da o verso un microcomputer.
Flag register (Registro di flag)	Un registro costituito da più flag.
General purpose register (Registro ad utilizzo generale o non specializzato)	Nel microprocessore 8080, i registri B, C, D, E, H ed L, anche se non è del tutto vero per la coppia H ed L.
Hexadecimal code (Codice esadecimale)	Un codice digitale a radice 16, che utilizza i numeri decimali da 0 a 9 e le lettere da A ad F per rappresentare i numeri da 10 a 15.
HI address byte (Byte di indirizzo HI)	Gli 8 bit più significativi in un indirizzo di memoria a 16 bit. Può essere abbreviato come HI o H.
Immediate addressing (Indirizzamento immediato)	Byte di dati che sono contenuti in un'istruzione multibyte.
Increment (Incrementare)	Aumentare il valore di una parola binaria. Tipicamente, aumentarne il valore di una unità.
Instruction (Istruzione)	Un set di caratteri che definiscono un'operazione unitamente ad uno o più indirizzi, oppure no, e che, come unità, determina l'attuazione da parte del microcomputer di una certa operazione sulle quantità indicate.[7]
Instruction code (Codice istruzione)	Un simbolo binario unico che codifica un'operazione eseguibile dal microcomputer. Anche codice operativo.
Instruction decoder	Un decoder facente parte della CPU, che decodifica il codice operativo, in una serie di azioni

(Instruction decoder)	eseguibili dal microcomputer.
Instruction register (Registro istruzione)	Il registro che contiene il codice operativo.
Jump (Salto)	1. Determinare la selezione della successiva istruzione da una specifica locazione di memoria non sequenziale all'attuale. 2. Scostamento dalla normale esecuzione sequenziale di un programma.
Label (Etichetta)	Uno o più caratteri che «danno il nome» ad un dato o alla locazione di memoria in cui è allocata una istruzione. La «label» è un insieme dei caratteri alfanumerici usuali, arbitrariamente definito.
LO address byte (Byte di indirizzo LO)	Gli 8 bit meno significativi in un indirizzo di memoria a 16 bit. Può essere abbreviato come LO o L.
Machine code (Codice macchina)	Un'istruzione scritta come sequenza di 0 ed 1 binari, specifica di quella sola istruzione. Rappresentazione binaria di un'istruzione.
Machine instruction (Istruzione macchina)	Vedi machine code (codice macchina).
Machine language (Linguaggio macchina)	Vedi machine code (codice macchina).
Mnemonic (Mnemonico)	Qualcosa utilizzato per aiutare la memoria umana a ricordare.
Mnemonic code (Codice memoria)	Istruzioni scritte in una forma atta ad essere ricordata facilmente, che però deve essere convertita in linguaggio macchina, per l'esecuzione.[4] Esempio: ADD, MPY, STO.
Mnemonic instructions (Istruzioni mnemoniche)	Vedi mnemonic code (codice mnemonico).
Mnemonic language (Linguaggio mnemonico)	Linguaggio di programmazione che si basa su simboli facilmente ricordabili, e che può successivamente essere assemblato in linguaggio macchina.[4]

Mnemonic operation code (Codice operativo mnemonico)	Vedi mnemonic code (codice mnemonico).
Mnemonic symbol (Simbolo mnemonico)	Un simbolo scelto in modo tale da essere facilmente ricordabile; ad esempio, l'abbreviazione MPY è usata per «moltiplicazione».
Octal code (Codice ottale)	Un codice digitale basato sulla radice 8, in cui i numeri decimali da 0 a 7 rappresentano 8 distinti stati.
Operand (Operando)	La quantità che è sottoposta a manipolazioni varie da parte del microcomputer.
Operation (Operazione)	Una specifica azione che un microcomputer realizzerà ogniqualvolta è richiesto da un'istruzione.[4]
Operation code (Codice operativo)	Vedi instruction code (codice istruzione).
Parity (Parità)	Un metodo per attuare il test della correttezza dei numeri binari. Un bit extra, detto parity bit, è sommato al numero. Se si usa la parità even (pari), la somma di tutti gli 1 del numero e del suo corrispondente bit di parità, deve essere «even». Se si usa la parità odd (dispari), la somma degli 1 deve essere «odd».
Parity flag (Flag di parità)	Un flip-flop tale per cui se la somma modulo 2 dei bit del risultato di una data operazione è 0, allora si setta ad 1.
Pop (Pop)	Anche «pull». Il recuperare dati dallo stack.
Program (Programma)	Vedi computer program (programma di un computer).
Program counter (Contatore di programma)	Registro a 16 bit che contiene l'indirizzo di memoria del successivo byte istruzione che deve essere eseguito in un programma.
Push (Push)	Collocare dati nello stack.
Register (Registro)	Un dispositivo digitale di memorizzazione la cui capacità è usualmente quella di una parola del microcomputer.
Register pair (Coppia di registri)	Nel microprocessore 8080, una coppia di registri general purpose che, insieme, realizzano una parola a 16 bit, trattata come unità. Le 3 coppie

per il microprocessore 8080 sono: B e C, D ed E, H ed L.

Register pair addressing (Indirizzamento relativo ad una coppia di registri)	Un byte di dati è acquisito tramite una istruzione ad un byte, che usa una coppia di registri, in genere H ed L, per generare l'indirizzo a 16 bit necessario.
Return (Ritorno)	Un tipo particolare di salto, in cui il processo riprende l'esecuzione di un main program, in corrispondenza del valore che il program counter aveva al momento in cui si era verificato il salto.
Return from subroutine (Ritorno da subroutine)	Vedere return (ritorno).
Routine (Routine)	Un set di istruzioni in data sequenza, atte a far eseguire al microcomputer un certo compito od una sequenza di operazioni. Alternativamente, una suddivisione di un programma consistente di due o più istruzioni, funzionalmente tra loro in relazione.
Sign flag (Flag di segno)	Un flip-flop che si pone ad 1 logico se il bit più significativo del risultato di una operazione vale 1.
Single byte instruction (Istruzione ad un solo byte)	Un'istruzione costituita da 8 bit contigui, occupanti una singola locazione di memoria.[5]
Source register (Registro sorgente)	Il registro che contiene una parola ad 8 bit che deve subire un trasferimento.
Stack	Un'area della memoria che memorizza in modo temporaneo i contenuti dei registri e gli indirizzi di ritorno dalla subroutine.
Stack pointer	Un registro a 16 bit che indica la locazione corrente dello stack.
Stack pointer addressing (Indirizzamento relativo allo stack pointer)	Due byte sono acquisiti tramite un'istruzione ad 1 byte che trasferisce il dato da un'area di memoria detta stack, verso una coppia di registri o verso il programma counter.
Subroutine	Una piccola porzione di programma che non fa parte della sequenza principale. Si entra nella subroutine solo tramite un'operazione di call.

Symbolic address (Indirizzo simbolico)	Detto anche floating address (indirizzo mobile). Si sceglie una label per identificare una particolare parola, funzione, od altre informazioni indipendenti dalla locazione dell'informazione nella routine.
Symbolic code (Codice simbolico)	Un codice tramite il quale vengono scritti i programmi, in linguaggio sorgente; cioè, ci si riferisce alla locazione di memorizzazione ed alle operazioni macchina con nomi simbolici ed indirizzi che non dipendono dai loro nomi ed indirizzi legati alla struttura hardware.[4]
Symbolic coding (Codifica simbolica)	Qualsiasi sistema di codifica che utilizza indirizzi simbolici al posto di quelli reali.
Symbolic language programming (Linguaggio di programmazione simbolico)	Vedere assembly language programming (linguaggio di programmazione assembler).
Symbolic programming (Programmazione simbolica)	Un programma utilizzante simboli al posto di numeri per le operazioni e le locazioni in un microcomputer. Per quanto la scrittura del programma sia più veloce e facile, un programma assemblatore deve poi essere usato per tradurre tutte le notazioni simboliche in linguaggio macchina.
Three-byte instruction (Istruzione a 3 byte)	Un'istruzione che è costituita da 24 bit contigui, occupanti 3 successive locazioni di memoria.
Two-byte instruction (Istruzione a 2 byte)	Una istruzione che è costituita da 16 bit contigui, occupante 2 successive locazioni di memoria.
Inconditional (Incondizionato)	Non soggetto a condizioni esterne, con riferimento al verificarsi, oppure no, di una data istruzione.
Inconditional jump (Salto non condizionato)	Una istruzione che interrompe il normale flusso sequenziale e che indica l'indirizzo da cui deve essere presa la successiva istruzione.
Inconditional return (Ritorno non condizionato)	Una istruzione di ritorno non soggetta a condizioni.

Word *(Parola)* — Un gruppo di 16 bit contigui occupanti due successive locazioni di memoria (questa definizione è data dalla Intel per il microprocessore 8080, ed in effetti si potrebbe non concordare, in quanto se il concetto di parola è legato a quello di parallelismo, se l'8080 ha parallelismo 8, allora la sua parola è 8 bit (N. D. T.)).

Zero flag *(Flag di zero)* — Un flip-flop che va allo stato logico 1, se il risultato di una istruzione è 000_8.

CHE COSA E' UN PROGRAMMA DI UN COMPUTER?

Graf ha dato questa definizione di *programma di un computer*[4]:

«Una serie di istruzioni o di frasi preparate in una forma accettabile per un computer, il cui scopo è di ottenere un certo risultato». Questa è una definizione accettabile, perché non specifica che cos'è il risultato desiderato. In alcuni casi, possiamo cercare di valutare un'equazione matematica, mentre in altri possiamo semplicemente cercare di risistemare i dati in ingresso in una forma più conveniente, che viene o memorizzata o fornita come uscita. Con i microprocessori, saremo sempre più interessati a scrivere programmi che controlleranno le operazioni di un dispositivo o di un gruppo di dispositivi. In una lavatrice domestica, per esempio, può darsi che vogliamo controllare la quantità d'acqua usata, la temperatura dell'acqua nei diversi cicli di lavaggio, il numero e il tipo di cicli usati per lavare un tipo particolare di tessuto, e la durata di ogni ciclo. Infine, possiamo chiedere al microprocessore di suonare un campanello o di emettere un sibilo quando il ciclo di lavaggio è stato completato.

CHE COSA E' UN'ISTRUZIONE?

Un'*istruzione* si può definire in questo modo[5]:

> Un insieme di caratteri che definiscono un'operazione insieme ad uno o più indirizzi, o senza indirizzi, e che, come unità, fa sì che il computer esegua l'operazione sulle quantità indicate.
>
> Del concetto di operazione parleremo più avanti.

Un *carattere* è[5]:

> Un simbolo appartenente ad un insieme di simboli elementari, come quelli che corrispondono ai tasti delle macchine da scrivere. I simboli comprendono di solito le cifre decimali da 0 a 9, le lettere da A a Z, i segni di punteggiatura, il dollaro, le virgole, i simboli delle operazioni e qualunque altro simbolo che un computer può leggere, memorizzare o scrivere.

Nella programmazione del computer, può capitare di usare tutta la tastiera della macchina da scrivere, compresi simboli quali @, #, $, %, ¢, &, °, (,), ?, /, e !.

Le istruzioni si presentano in varie forme. Possono essere *numeri binari:*

10110001_2
00011100_2
11101001_2

numeri ottali:

027_8
353_8
124_8
001_8

numeri esadecimali

OA_{16}
79_{16}
FF_{16}
$D3_{16}$
BE_{16}

numeri decimali

10
35
26
05

codici mnemonici

NOP
MOV B, C
INR H
ADD D
SUB L
OUT
HLT
JMP

parole complete

ADDIZIONE
SOTTRAZIONE
CONFRONTO
NESSUNA OPERAZIONE
ALT
SALTO
RICHIAMO DI SUBROUTINE
RIENTRO DALLA SUBROUTINE
OR-ESCLUSIVO

o intere espressioni matematiche

X = A**2 + B*Y + C
X = SQRT (B**2 — 4*A*C)

per menzionare solo alcuni dei tipi più comuni.

CHE COSA E' UN'OPERAZIONE?

Ritorniamo a Graf per una semplice definizione del termine «*operazione*»[4]:

> Un'operazione specifica che un computer eseguirà ogniqualvolta un'istruzione la richiamerà (es. addizione, divisione).

Il numero di operazioni diverse che un computer può eseguire e la velocità alla quale tali operazioni vengono eseguite fornisce una misura della «potenza» del computer stesso. Le operazioni associate con il microprocessore 8080 comprendono:

Operazioni di trasferimento delle informazioni	Trasferire le informazioni dall'accumulatore alla memoria
	Trasferire le informazioni dall'accumulatore al registro
	Trasferire le informazioni dalla memoria all'accumulatore
	Trasferire le informazioni dal registro all'accumulatore
	Trasferire le informazioni nello stack
	Trasferire le informazioni dallo stack
	Trasferire le informazioni nello stack pointer
	Trasferire le informazioni nel contatore di programma
Operazioni aritmetiche	Sommare all'accumulatore
	Sottrarre dall'accumulatore
	Confrontare con l'accumulatore
	Far ruotare l'accumulatore verso sinistra
	Far ruotare l'accumulatore verso destra
Operazioni logiche	AND con l'accumulatore
	OR con l'accumulatore
	OR Esclusivo con l'accumulatore
Operazioni della subroutine	Chiamare la subroutine
	Chiamare la subroutine se il flag è a livello logico 0
	Chiamare la subroutine se il flag è a livello logico 1
	Rientrare dalla subroutine

	Rientrare dalla subroutine se il flag è a livello logico 0 Rientrare dalla subroutine se il flag è a livello logico 1 Riprendere da una locazione di subroutine specificata
Operazioni di ingresso/ uscita	Inserire i dati da un dispositivo nell'accumulatore Mettere in uscita i dati dell'accumulatore verso un dispositivo
Operazioni di incremento/ decremento	Incrementare i contenuti dell'accumulatore Decrementare i contenuti dell'accumulatore Incrementare i contenuti del registro Decrementare i contenuti del registro Incrementare i contenuti della memoria Decrementare i contenuti della memoria
Operazioni di salto	Saltare incondizionatamente Saltare ad una locazione di memoria se il flag è a livello logico 0 Saltare ad una locazione di memoria se il flag è a livello logico 1
Altre operazioni	Complementare l'accumulatore Complementare il flip-flop di carry Posizionare il flip-flop di carry Aggiustamento decimale dell'accumulatore Abilitare le interruzioni Disabilitare le interruzioni Nessuna operazione Alt

LINGUAGGIO MACCHINA

Il moderno computer elettronico digitale è in grado di eseguire manipolazioni usando solo segnali elettronici binari, di solito a 0 V (livello logico 0) e + 5 V (livello logico 1). Così, *ogni istruzione è scritta con una sequenza di 1 e di 0 che caratterizzano specificatamente quell'istruzione e nessun'altra.* Questa rappresentazione binaria dell'istruzione di un computer si chiama *linguaggio macchina o codice macchina.* Per esempio, l'istruzione il linguaggio macchina 00000111_2 fa ruotare i contenuti dell'accumulatore di un bit verso sinistra, nel microprocessore 8080. L'istruzione 00001111_2 fa ruotare i contenuti dell'accumulatore di un bit verso

destra. Nell'insieme, è possibile creare $2^8 = 256$ diverse istruzioni binarie a 8 bit in linguaggio macchina per un microprocessore a 8 bit come l'8080.

Di queste 256 istruzioni possibili, ne esiste un totale di 244. Ecco ancora alcuni esempi delle istruzioni in linguaggio macchina per l'8080.

00000000	Nessuna operazione
00000100	Incrementare i contenuti del registro B di uno
00000101	Decrementare i contenuti del registro B di uno
00001100	Incrementare i contenuti del registro C di uno
00010100	Incrementare i contenuti del registro D di uno
00011100	Incrementare i contenuti del registro E di uno
00110111	Settare il flip-flop di carry al livello logico 1
00111111	Complementare il flip-flop di carry
01000111	Trasferire i contenuti dell'accumulatore nel registro B
01011000	Trasferire i contenuti del registro B nel registro E
10000000	Sommare i contenuti del registro B ai contenuti dell'accumulatore
10010000	Sottrarre i contenuti del registro B dall'accumulatore
10110000	Eseguire un'operazione OR dei contenuti del registro B con i contenuti dell'accumulatore
11001001	Rientrare dalla subroutine
11010111	Chiamare la subroutine all'indirizzo di memoria $H = 000_8$ e $L = 020_8$
11111011	Abilitare l'interruzione

Tutti i codici macchina elencati sono codici binari.

CODICI MACCHINA OTTALI ED ESADECIMALI

Può essere difficile ricordare le istruzioni in codice macchina binario a 8 bit, perciò coloro che programmano in codice macchina spesso convertono tali istruzioni in *codice ottale o esadecimale.* Di tali codici abbiamo parlato nel Capitolo 5 del *Bugbook 1,* ma converrà ripeterli ora.

Le cifre ottali per gli otto numeri binari a tre bit che vanno da 000 a 111 sono:

Cifra ottale	*Numero binario*
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Il microprocessore 8080 genera istruzioni in codice macchina bi-

nario a 8 bit, perciò dobbiamo essere in grado di convertire una parola binaria a 8 bit in codice ottale. Il processo tramite il quale arriviamo a questo richiede tre fasi:

1. Scrivere le parole binarie a 8 bit per intero: XXXXXXXX.
2. Dividere la parola binaria a 8 bit in gruppi di tre, iniziando dal bit meno significativo. Vengono creati due gruppi di tre e un gruppo di due, come in XX XXX XXX.
3. Sostituire da 0 a 7 in ognuno dei gruppi per ottenere la parola in codice ottale finale.

Come esempio, supponiamo di avere la parola binaria 10011011. Dapprima la dividiamo in gruppi:

$$10011011_2 = 10\ 011\ 011$$

Poi, scriviamo la parola ottale equivalente per ognuno di questi gruppi:

$$10\ 011\ 011 = 233_8$$

per produrre 233_8, l'ottale equivalente della parola binaria 10011011. Notate che il numero ottale più significativo ha solo due bit binari, e non può essere maggiore dell'ottale 3. Riassumendo, la conversione di una parola binaria a 8 bit nel suo equivalente ottale avviene in questo modo:

$$XXXXXXXX_2 = XX\ XXX\ XXX$$

Cifra ottale a 2 bit (XX) — *Cifra ottale a 3 bit* (XXX) — *Cifra ottale a 3 bit* (XXX)

Il codice esadecimale per i sedici numeri binari a 4 bit è il seguente:

Cifra esadecimale	*Numero binario*
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Per convertire una parola binaria a 8 bit in esadecimale, dobbiamo:

1. Scrivere per intero la parola binaria a 8 bit, XXXXXXXX.
2. Suddividere la parola binaria a 8 bit in due gruppi di quattro bit, come in XXXX XXXX.
3. Sostituire da 0 a F per ogni gruppo per ottenere la parola finale in codice esadecimale.

Possiamo, ad esempio, convertire 10011011 in esadecimale. Dapprima lo dividiamo in due gruppi di quattro bit:

$$10011011_2 = 1001\ 1011$$

Poi, scriviamo la parola esadecimale equivalente per ognuno di questi gruppi:

$$1001\ 1011 = 9B_{16}$$

per produrre $9B_{16}$, l'equivalente esadecimale della parola binaria originale. Riassumendo, la conversione di una parola binaria a 8 bit nel suo equivalente esadecimale avviene in questo modo:

$$XXXXXXXX_2 = XXXX\ XXXX$$

Cifra esadecimale a 4 bit *Cifra esadecimale a 4 bit*

Con il microprocessore 8080, l'indirizzo di memoria contiene 16 bit. Una parola di questo genere può anche essere convertita o in codice ottale o in codice esadecimale. Per esempio, la locazione di memoria 1001100111000101_2 può essere scritta così:

$$1001100111000101_2 = 1\ 001\ 100\ 111\ 000\ 101_2 = 114705_8$$
$$= 1001\ 1001\ 1100\ 0101_2 = 99C5_{16}$$

Comunque, quando programmiamo in codice ottale per l'8080, vedremo spesso un indirizzo a 16 bit che viene suddiviso in due byte di otto bit. In questo caso, gli otto bit più significativi sono il *byte d'indirizzo HI* e gli otto bit meno significativi sono il *byte d'indirizzo LO.* Quando è così suddiviso, ogni byte viene trattato come un numero ottale separato che varia da 000 a 377. Perciò il nostro indirizzo può essere scritto così:

$$1001100111000101_2$$

10 011 001	11 000 101
2 3 1	3 0 5

Un indirizzo di questo tipo ha un byte d'indirizzo HI di 231_8 ed un byte d'indirizzo LO di 305_8.

La tabella 3-1 mostra delle conversioni utili di numeri binari, ottali ed esadecimali.

Le istruzioni in linguaggio macchina elencate nel paragrafo «Linguaggio Macchina» possono quindi essere riscritte nel modo seguente:

Codice macchina binario	*Codice macchina esadecimale*	*Codice macchina ottale*
00000000	00	000
00000100	04	004
00000101	05	005
00001100	0C	014
00010100	14	024
00011100	1C	034
00110111	37	067
00111111	3F	077
01000111	47	107
01011000	58	130
10000000	80	200
10010000	90	220
10110000	B0	260
11001001	C9	311
11010111	D7	327
11111011	FB	373

In questo libro, *scriveremo le istruzioni macchina in codice ottale.* Vi sono molte ragioni per cui lo facciamo:

- E' più facile convertire un numero ottale in numero binario che un numero esadecimale in numero binario. I simboli esadecimali A, B, C, D, E, F, possono all'occasione creare confusione.
- I display a LED a sette segmenti sono più comuni e meno costosi dei display a LED esadecimali.
- *Le origini delle istruzioni specifiche del microprocessore Intel 8080 sono più facili da capire se tali istruzioni sono scritte in codice ottale anziché in codice esadecimale.*

Questi tre vantaggi compensano il fatto che vi sono solo due cifre nell'istruzione esadecimale, contro le tre cifre dell'istruzione ottale.

CODICE MNEMONICO

Mnemonico è un termine che viene usato per descrivere qualcosa che serve ad assistere la memoria umana. Tenendo presente questo concetto, facciamo seguire le definizioni:

Tabella 3-1: Conversioni binarie, ottali, esadecimali, decimali

Numero Binario	Numero Ottale	Numero Esadecimale	Numero Decimale
00000000	000	00	0
00000001	001	01	1
00000010	002	02	2
00000011	003	03	3
00000100	004	04	4
00000101	005	05	5
00000110	006	06	6
00000111	007	07	7
00001000	010	08	8
00001001	011	09	9
00001010	012	0A	10
00001011	013	0B	11
00001100	014	0C	12
00001101	015	0D	13
00001110	016	0E	14
00001111	017	0F	15
00010000	020	10	16
00011000	030	18	24
00100000	040	20	32
00101000	050	28	40
00110000	060	30	48
00111000	070	38	56
01000000	100	40	64
01001000	110	48	72
01010000	120	50	80
01011000	130	58	88
01100000	140	60	96
01101000	150	68	104
01110000	160	70	112
01111000	170	78	120
10000000	200	80	128
10001000	210	88	136
10010000	220	90	144
10011000	230	98	152
10100000	240	A0	160
10101000	250	A8	168
10110000	260	B0	176
10111000	270	B8	184
11000000	300	C0	192
11001000	310	C8	200
11010000	320	D0	208
11011000	330	D8	216
11100000	340	E0	224
11101000	350	E8	232
11110000	360	F0	240
11111000	370	F8	248
11111001	371	F9	249
11111010	372	FA	250
11111011	373	FB	251
11111100	374	FC	252
11111101	375	FD	253
11111110	376	FE	254
11111111	377	FF	255

Mnemonic code (Codice mnemonico)	Istruzioni scritte in una forma che il programmatore può convertire più tardi in linguaggio macchina da un computer.
Mnemonic language (Linguaggio mnemonico)	Un linguaggio di programmazione basato su simboli facili da ricordare e che può essere assemblato in linguaggio macchina da un computer.
Mnemonic operation code; mnemonic instruction (Codice operativo mnemonico; istruzioni mnemoniche)	Istruzioni scritte secondo una notazione esplicita come ADD, MPY, STO.[4]
Mnemonic symbol (Simbolo mnemonico)	Un simbolo scelto in modo che sia di aiuto alla memoria umana; per es. l'abbreviazione MPY di «multiply».

Si possono scrivere parecchie istruzioni mnemoniche diverse per le 244 diverse istruzioni in linguaggio macchina dell'8080. Dato che crediamo nella standardizzazione, useremo queste istruzioni mnemoniche, suggerite dalla Intel Corporation. Tali istruzioni vengono fornite in diversi testi [4] dalla Intel, fra i quali:

- *Intel 8080 Microcomputer System Manual,* Settembre 1975;
- *Intel Assembly Language Programming Manual,* 1976 (Revisione C);
- Intel Assembly Language Reference Card, 1974

nonché in pratiche schede in codice, come le seguenti, fornite dalla Tychon, Inc.:[13]

- Tychon, Inc., 8080 Octal and Hexadecimal Code Cards.

Le istruzioni in linguaggio macchina elencate sia nel paragrafo «Linguaggio macchina» che in quello «Codici ottali ed esadecimali», possono quindi essere scritte in questo modo:

Codice macchina ottale	*Codice mnemonico*
000	NOP
004	INR B
005	DCR B
014	INR C
024	INR D
034	INR E
067	STC
077	CMC

107	MOV B, A
130	MOV E, B
200	ADD B
220	SUB B
260	ORA B
311	RET
327	RST 2
373	EI

COME IMPARARE A PROGRAMMARE UN COMPUTER?

Fra coloro che leggeranno questo capitolo, alcuni sapranno già programmare in *linguaggio macchina,* e forse desiderano solo arrivare in fretta al set di istruzioni del microprocessore 8080. Molti altri si troveranno a programmare e ad interfacciare un computer per la prima volta. Potrebbe venirvi in mente una domanda importante: *come fare per imparare a programmare il computer?* Nei paragrafi che seguono, vi vengono dati dei suggerimenti in proposito. Tali commenti vengono forniti all'inizio di questo capitolo, e non alla fine, perché può esservi utile avere una «cartina stradale» del percorso che state compiendo con questo libro. Può darsi che all'inizio i termini e i concetti che userete in questo paragrafo non vi siano familiari. La prima volta, leggete questa parte in fretta, ritornerete poi indietro quando ne avrete bisogno.

Dove Dobbiamo Arrivare?

In questo libro, imparerete a creare un'interfaccia fra i microcomputer e semplici circuiti digitali che consistono di solito di chip della serie 7400 e di pochi chip specializzati della serie 8000. Dovrete imparare come trasmettere le informazioni fra il microcomputer e i dispositivi esterni di ingresso/uscita, compresi i circuiti integrati. Così, vi mostriamo due delle istruzioni più importanti che dovreste imparare, e che generano impulsi di selezione dispositivi:

323 <B2> Genera un impulso di selezione dispositivo per sincronizzare l'uscita degli otto bit dei dati dell'accumulatore al dispositivo che ha il codice dispositivo dato nel secondo byte dell'istruzione.

e

333 <B2> Genera un impulso di selezione dispositivo per sincronizzare l'ingresso degli otto bit di dati nell'accumulatore dal dispositivo con il codice dispositivo dato nel secondo byte dell'istruzione.

Queste sono le istruzioni *OUT* e *IN,* rispettivamente. Sicuramente, userete spesso l'istruzione OUT nei capitoli successivi.

In un capitolo seguente, imparerete come si scrivono le *subrou-*

tine. Con tali subroutine, il computer salta da un programma principale a qualche altra posizione di memoria, dove inizia ad eseguire la subroutine. Una volta completata la subroutine, esso ritorna al programma principale. Il computer, quindi, deve conoscere la posizione del programma principale a cui dovrebbe ritornare. Viene aiutato in questo da un gruppo di posizioni di memoria, conosciute come *stack*, che memorizzano gli indirizzi di rientro per le subroutine. *Non sarete in grado di usare nessuna subroutine, comprese quelle che si trovano sul chip di una PROM programmata, a meno che non posizioniate lo stack della memoria di lettura/scrittura che avete in quel momento nel vostro microcomputer.* Come riposizionare lo stack? Si usano le istruzioni a tre byte seguenti:

061	Riposiziona lo stack pointer all'indirizzo di 16 bit dato dai due byte seguenti di questa istruzione.
<B2>	Byte d'indirizzo di memoria LO
<B3>	Byte d'indirizzo di memoria HI

Una sera uno degli autori sprecò molte ore a tentare di scrivere una subroutine prima di aver capito che era necessario riposizionare lo stack da dove si trovava inizialmente (e cioè, nel caso vi interessasse, HI = 377_8 e LO = 377_8). Le subroutine sono una ginnastica molto importante per la programmazione, così importante che è una buona idea nominare nel modo giusto lo stack all'inizio di questo capitolo.

In generale, non incomincerete l'esecuzione del programma alla locazione di memoria HI = 000_8 e LO = 000_8 e non proseguirete sequenzialmente attraverso la memoria che avete a disposizione. Salterete tutt'attorno, qualche volta in avanti, qualche volta all'indietro. In questo modo, avrete bisogno di sapere come saltare da un punto del vostro programma, in un altro, dove l'esecuzione continua. L'istruzione di cui avrete bisogno è il *Salto Incondizionato,* un'istruzione a tre byte:

303	Salta incondizionatamente alla locazione di memoria a 16 bit data dai due byte seguenti di questa istruzione.
<B2>	Byte d'indirizzo di memoria LO
<B3>	Byte d'indirizzo di memoria HI

E' una istruzione notevole. Potete fare meraviglie usandola.

Può darsi che vogliate chiamare una subroutine, il che significa semplicemente che chiedete al microcomputer di memorizzare l'indirizzo di memoria dell'istruzione seguente del programma principale, e poi di saltare a qualche altro indirizzo di memoria dove inizia l'esecuzione della subroutine. Per fare questo, vi servirà l'istruzione di *Chiamata Incondizionata,* anch'essa un'istruzione a tre byte:

315 Chiama incondizionatamente la subroutine posta alla

locazione di memoria a 16 bit data dai due byte seguenti di questa istruzione.
<B2> Byte d'indirizzo di memoria LO
<B3> Byte d'indirizzo di memoria HI

Questa istruzione, comunque, non è sufficiente. Una volta eseguita la subroutine, vi servirà un'istruzione che vi riporti al programma principale. Questa è l'istruzione di *Rientro Incondizionato,* un'istruzione ad un byte:

311 Rientra incondizionatamente al programma principale. L'indirizzo specifico del programma principale è contenuto in due locazioni di memoria nello stack.

Abbiamo usato il termine *«incondizionato»* in tutte e tre le istruzioni di questa pagina. Esiste anche un certo numero di istruzioni di rientro, di chiamata e di salto condizionato che eseguono le loro azioni indicate solo se il flag ha uno specifico livello logico. Questo libro non ne userà molte, perciò dovrete impararle da soli. Probabilmente la sola istruzione condizionata che userete spesso in questo libro è un'istruzione associata alla programmazione dei loop di timing:

302 Salta alla locazione di memoria a 16 bit data dai due byte seguenti di questa istruzione se il flag di zero è a livello logico 0.
<B2> Byte d'indirizzo di memoria LO
<B3> Byte d'indirizzo di memoria HI

Vi chiederete: che cos'è un *flag di zero?* E' semplicemente un flipflop che va ad un livello logico 1 se i contenuti del registro o della memoria su cui si è operato nell'istruzione precedente vanno a 000_8. Questa è un'istruzione importante e la userete spesso nel Capitolo 5.

Osservate che non abbiamo ancora parlato delle operazioni logico/aritmetiche di somma, sottrazione, And, Or e di altri tipi. La ragione è che non userete molte di queste istruzioni in questo Bugbook. Probabilmente sarete troppo occupati a mettere in uscita e ad inserire dati per preoccuparvi di sommare una coppia di numeri. C'è un'istruzione che piace molto agli autori:

227 Sottrarre i contenuti dell'accumulatore dai contenuti dell'accumulatore, cioè azzerare l'accumulatore.

Con questa istruzione, settate il valore dell'accumulatore a 000. E' molto utile.

Quando effettuate i loop di timing, può darsi che vogliate incrementare i registri universali B e C. Le istruzioni che vi serviranno sono:

004 Incrementa il registro B di uno.
014 Incrementa il registro C di uno.

Vorrete anche *decrementare* questi due registri:

005 Decrementa il registro B di uno.
015 Decrementa il registro C di uno.

Vorrete certamente arrestare il computer dopo che ha eseguito un programma. Per farlo, userete la nota e rispettata istruzione *Halt:*

166 Arresta il microcomputer.

Se siete pigri, potete programmare il computer per non fare niente:

000 Nessuna operazione.

Questa istruzione non fa assolutamente niente. All'interno del programma, consuma solo del tempo, precisamente 2 μs. Potete usarla per fornire degli spazi nel vostro programma che potete riempire in un secondo tempo. Può darsi che vogliate azzerare un registro, come accadrebbe con un loop di timing lungo. Prima, potete azzerare l'accumulatore. Poi trasferite i contenuti dell'accumulatore nel registro desiderato, come nelle seguenti istruzioni:

107 Trasferisci i contenuti dell'accumulatore ne! registro B.
117 Trasferisci i contenuti dell'accumulatore nel registro C.
127 Trasferisci i contenuti dell'accumulatore nel registro D.

Può darsi che vogliate caricare uno specifico numero binario a 8 bit nell'accumulatore o nel registro. Userete a questo scopo le seguenti istruzioni immediate a due byte:

006 Trasferisci i dati del secondo byte di questa istruzione
<B2> nel registro B.
026 Trasferisci i dati del secondo byte di questa istruzione
<B2> nel registro D.
076 Trasferisci i dati del secondo byte di questa istruzione
<B2> nell'accumulatore.

Probabilmente vorrete caricare una coppia di numeri binari a 8 bit in due registri con la stessa istruzione. Eccone un esempio:

041 Trasferisci i dati del secondo e del terzo byte di questa istruzione nella coppia di registri H.
<B2> Questo byte va nel registro L
<B3> Questo byte va nel registro H

Nel corso del Capitolo 8, avrete bisogno di abilitare il *flag di interrupt* in modo da poter interrompere il microcomputer dal pannello frontale. Ciò avviene per mezzo dell'istruzione seguente:

373 Abilita il sistema d'interruzione, dopo l'esecuzione dell'istruzione successiva.

Nel Capitolo 8, forzeremo un'istruzione ad un byte durante l'interruzione. Questa è un'istruzione che richiama una subroutine ad uno

degli indirizzi di memoria seguenti (tutti con HI = 000_8): 000_8, 010_8, 020_8, 030_8, 040_8, 050_8, 060_8 o 070_8. Queste sono le istruzioni di ripristino, di cui vi diamo un esempio:

317 Richiama la subroutine all'indirizzo di memoria dato da H = 000_8 e L = 010_8.

Potete sbizzarrirvi con l'accumulatore e il flag di carry. Per esempio; potete:

007 Far ruotare i contenuti dell'accumulatore di una posizione verso sinistra.
017 Far ruotare i contenuti dell'accumulatore di una posizione verso destra.
047 Regolare l'accumulatore secondo il sistema decimale.
057 Complementare l'accumulatore.
067 Settare il flag di carry a livello logico 1.
077 Complementare il flag di carry.

E, infine, potete trasferire i contenuti di una locazione di memoria specificata nell'accumulatore, come nella seguente istruzione:

176 Trasferisci i contenuti della locazione di memoria indirizzata dalla coppia di registri H, L nell'accumulatore.

Veniamo ora al punto di questo paragrafo: *le istruzioni suddette sono quasi tutto quello di cui avrete bisogno per questo libro!* L'enfasi viene posta sull'interfacciamento del computer, non sulla programmazione. Sono stati forniti passi di programma sufficienti per permettervi di far fare qualcosa di utile al vostro circuito integrato interfacciato.

Come Imparare a Programmare un Microcomputer?

Potete imparare a programmare il computer con l'aiuto di questo Bugbook e di alcuni esperimenti manuali con un microcomputer basato sull'8080. E' possibile fare un ottimo uso di questo libro, ma il fatto di avere prima un microcomputer renderebbe il vostro studio molto più interessante. Ecco alcuni suggerimenti:

- Come primo passo nell'apprendimento della programmazione, osservate il set di istruzioni del microcomputer e tentate di imparare quello che potete.

  In questo libro il set di istruzioni dell'8080 è stato organizzato in vari modi per aiutarvi: (1) un elenco mnemonico ottale/esadecimale di tutte le 256 istruzioni, da 000_8 a 377_8; (2) un elenco alfabetico delle istruzioni mnemoniche; (3) un elenco che raggruppa le istruzioni, compresi il gruppo di trasferimento dei dati, il gruppo aritmetico, il gruppo logico, il gruppo di salto e il gruppo di controllo stack/ingresso/uscita/macchina; (4) un sommario delle istruzioni su un solo foglio; (5) descrizioni dettagliate delle diverse classi di istruzioni, dove ogni classe o gruppo viene a seconda di come viene decodificato dal decodificatore di istruzioni *all'interno del chip 8080.*

- Studiate i numerosi esempi di programmazione dati in questo libro.

  Non abbiamo usato tutte le istruzioni dell'8080, ma abbiamo usato molte delle più importanti. Dovreste imparare da questo libro che, per essere utile, un programma non deve essere lungo e involuto. Vedrete come generare impulsi di selezione dispositivi ed attuare un latch sui dati dell'accumulatore solo con poche istruzioni. Con molte istruzioni in più, potete generare un loop di timing.

- Fate pratica di programmazione scrivendo molti programmi semplici, ognuno dei quali illustra le caratteristiche di una diversa istruzione dell'8080.

  In questo libro, gli autori hanno seguito questa regola. Hanno tentato di mettere in risalto il comportamento di un numero limitato di istruzioni utili. Sperano di averlo fatto andando abbastanza a fondo e con chiarezza sufficiente perché voi siate in grado di seguirne l'esempio. Imparando le caratteristiche di altre istruzioni, vorrebbero incoraggiarvi a seguire le stesse procedure: ad esempio, per le istruzioni DAD, ANA, XRA, ORA, CMP; i salti condizionati, i richiami e i ritorni; e istruzioni interessanti quali PCHL, XTHL, XCHG, SPHL, POP e PUSH.

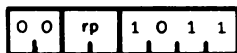
- Fate tutto quello che vi abbiamo detto con un microcomputer 8080 davanti a voi.

  Un microcomputer è un gran divertimento. Non renderete giustizia a questa nuova entusiasmante tecnologia leggendo semplicemente questo libro. L'esperienza manuale su di un computer vi darà l'approfondimento di esperienza che vi è necessario.

BIT, BYTE, PAROLA E INDIRIZZO

I termini *bit, byte, parola e indirizzo* sono talmente importanti che, sebbene ne abbiamo già dato la definizione nel capitolo precedente, lo rifaremo di nuovo qui. Le definizioni che seguono sono tratte dall'Assembly Language Programming Manual dell'Intel 8080 (Riferimento 7).

Bit *(Bit)* — La più piccola unità d'informazione che si possa rappresentare. Un bit può trovarsi in uno o due stati, rappresentati dalle cifre binarie 0 o 1.[7]

Byte *(Byte)* — Un gruppo di otto bit contigui che occupano una sola locazione di memoria.[4]

Una rappresentazione di un byte in memoria. I bit, hanno valore fisso o 1, sono indicati da 0 o 1; i bit che possono essere o 0 o 1 in circostanze diverse, sono rappresentati da lettere; perciò rp rappresenta un campo a 2 bit che contiene una delle otto possibili combinazioni di zeri e di uno.[7]

Word (Parola)	Un gruppo di 16 bit contigui che occupano due posizioni di memoria successive.[7]
Address (Indirizzo)	Un numero a 16 bit assegnato ad una posizione di memoria che corrisponde alla sua posizione sequenziale.[7]
Instruction (Istruzione)	L'operazione singola più piccola che il computer può essere diretto ad eseguire.[7]
Program (Programma)	Una sequenza di istruzioni che, prese come gruppo, permettono al computer di eseguire un compito desiderato.[7]
Field (Campo)	Un gruppo di bit in un byte o in una parola che viene trattato come una singola unità di informazione. Di solito il numero di bit nel campo è specificato, come, ad esempio, un campo a tre bit.

Ricordatevi che queste definizioni di byte, parola e indirizzo si riferiscono specificatamente al microprocessore 8080. Altri computer possono avere byte, parole e indirizzi più piccoli o più grandi.

ISTRUZIONI A PIU' BYTE

Il microprocessore 8080 può eseguire 244 istruzioni diverse. Molte di queste istruzioni si assomigliano, ed è stato detto che l'8080 ha solo 78 istruzioni realmente diverse. Non siamo preparati a discutere su questo fatto. Il punto è che, di 244 istruzioni, 200 sono istruzioni ad *un solo byte,* 18 sono istruzioni a *due byte,* e 26 sono istruzioni a *tre byte.* Questi tre termini si possono definire nel modo seguente:

Single-byte instruction (Istruzione ad un solo byte)	Un'istruzione che consiste di otto bit contigui che occupano una sola locazione di memoria.
Three-byte instruction (Istruzione a tre byte)	Un'istruzione che consiste di 24 bit contigui che occupano tre locazioni di memoria successive.
Two-byte instruction (Istruzione a due byte)	Un'istruzione che consiste di 16 bit contigui che occupano due locazioni di memoria successive.

Il primo byte di un'istruzione a più byte si chiama *codice operativo.* Il byte o i due byte rimanenti sono o un *byte di dati* a 8 bit, due *byte di dati* a 8 bit, un *codice dispositivo* a 8 bit, oppure un indirizzo di memoria a 16 bit che consiste di due *byte d'indirizzo* a 8 bit. Ecco le definizioni di questi termini nuovi:

Operation code, instruction code (Codice operativo, codice istruzioni)	Il codice a 8 bit per l'azione specifica che il microprocessore eseguirà.
Data byte (Byte di dati)	Il numero binario a 8 bit che il microprocessore 8080 userà in un'operazione aritmetica o logica o immagazzinerà in memoria o in un registro.
Device code (Codice dispositivo)	Il codice a 8 bit che serve per un dispositivo specifico di ingresso o di uscita. Questo codice viene decodificato da decodificatori esterni che, insieme all'impulso $\overline{\text{IN}}$ o $\overline{\text{OUT}}$ del microprocessore 8080, generano un solo impulso di selezione dispositivi.
HI address byte (Byte d'indirizzo HI)	Gli otto bit più significativi nella parola d'indirizzo di memoria a 16 bit per il microprocessore 8080. Abbreviato con H o HI.
LO address byte (Byte d'indirizzo LO)	Gli otto bit meno significativi nella parola d'indirizzo di memoria a 16 bit per il microprocessore 8080. Abbreviato con L o LO.

Chiaramente, una parola a 8 bit memorizzata in memoria può essere un codice operazioni, un byte di dati, un codice dispositivi, un byte d'indirizzo HI o un byte d'indirizzo LO. Come fate a dirlo? Esaminate ogni byte a 8 bit nel contesto dell'attuale programma del microcomputer. A meno che non vi sia un errore nel programma, il computer è in grado di distinguere facilmente fra i vari tipi di byte. Non ci saranno ambiguità. Tratteremo ancora questo argomento nel paragrafo seguente.

Il primo, secondo e terzo byte in un'istruzione a più byte sono rappresentati dai simboli <B1>, <B2> e <B3>, rispettivamente. Questo è quanto scrive la Intel Corporation e che noi seguiremo.

Le diciotto istruzioni a due byte per il microprocessore 8080 hanno i seguenti codici mnemonici:

otto MVI istruzioni: MVI B, MVI C, etc.
ADI
ACI
OUT
SUI
IN
SBI
ANI
XRI
ORI
CPI

Le 26 istruzioni a tre byte per l'8080 hanno i seguenti codici mnemonici:

quattro istruzioni LXI
nove istruzioni JUMP
nove istruzioni CALL SUBROUTINE
SHLD
LHLD
STA
LDA

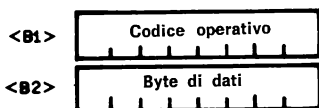
Useremo il codice mnemonico della Intel Corporation in tutto questo Bugbook.

Nella Figura 3-1, potete vedere delle semplici rappresentazioni di istruzioni ad un solo byte, a due e a tre byte.

ISTRUZIONI O DATI: COME FA IL COMPUTER A SAPERLO?

Con il microcomputer 8080, memorizziamo sia le istruzioni che i «dati», come quelli provenienti da uno strumento di laboratorio, nella stessa memoria. Le istruzioni e i dati possono quasi esistere fianco a fianco. C'è quindi ragione di chiedersi, come fa il microcomputer a sapere che differenza c'è fra i due?

La risposta di base è che le istruzioni in memoria vengono memorizzate come un blocco, o forse come un gruppo di blocchi collocati variamente attraverso la memoria, e lo stesso vale per i

(A) Un'istruzione ad un solo byte, che consiste solo di un codice operativo a 8 bit.

(B) Un'istruzione a due byte, che consiste di un codice operativo a 8 bit e di un byte di dati a 8 bit.

(C) Un'istruzione di ingresso e uscita a due byte, che consiste di un codice operativo a 8 bit e di un codice dispositivo a 8 bit.

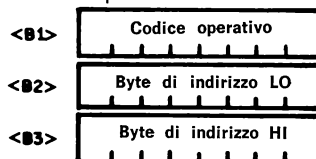
(D) Un'istruzione a tre byte, che consiste di un codice operativo a 8 bit e di due byte di dati a 8 bit.

(E) Un'istruzione a tre byte, che consiste di un codice operativo a 8 bit e di una parola indirizzi a 16 bit, suddivisa in un byte d'indirizzo LO e in un byte d'indirizzo HI.

Fig. 3-1. Rappresentazione delle istruzioni.

dati di uno strumento. Eccezione fatta per le istruzioni di tipo immediato, raramente vi trovate ad avere istruzioni e dati mescolati insieme. Tale miscuglio costituisce un cattivo modo di programmare ed una cattiva organizzazione di memoria. Perciò, il computer saprà sempre che sta operando sulle istruzioni. Se inizia nella giusta posizione di memoria ed il programma è scritto correttamente. Le istruzioni esisteranno sempre come un blocco coesivo in memoria. Quelle sparse nella memoria saranno istruzioni di salto ben definite, di richiamo, di rientro e di ripristino, che collegano le subroutine e i programmi sussidiari con il programma principale.

Per quanto concerne i dati in memoria, suggeriamo quanto segue: *salvo diversa indicazione, come nel caso di programmazioni apposite, il computer tratterà i dati in memoria come istruzioni e farà di tutto per cercare di eseguirle.* In altre parole, un programma scritto in maniera corretta avrà accesso ai dati e li memorizzerà tramite comandi per l'indirizzamento in memoria, quali MOV. Comunque, se per qualunque ragione il computer si trova ad operare in un blocco di dati durante l'esecuzione di un programma, esso tratterà i byte di dati come byte di istruzioni.

Come fa il computer a conoscere la differenza fra un'istruzione e un dato? Voi scrivete il programma in un modo tale, che il computer conosce la differenza. La mancanza di attenzione da parte vostra farà andare a rotoli il lavoro del computer. I programmi devono iniziare sempre con un'operazione valida, non con dei dati!

I REGISTRI DEL MICROPROCESSORE 8080

Il termine «registro» si può così definire:

Register (Registro)	Un dispositivo digitale di memorizzazione, la cui capacità è usualmente di una parola del microcomputer.

Un solo registro del microprocessore 8080 memorizza un solo byte, cioè otto bit contigui.

Nel chip dell'8080 vi sono due diversi set di registri: quelli che possiamo indirizzare da programma e quelli che non possiamo. I registri indirizzabili da programma sono mostrati nella Figura 3-2:

- Sei registri universali a 8 bit indirizzati singolarmente o a coppie:
  registro B
  registro C
  registro D
  registro E
  registro H
  registro L
- L'accumulatore a 8 bit, conosciuto anche come *registro A.*

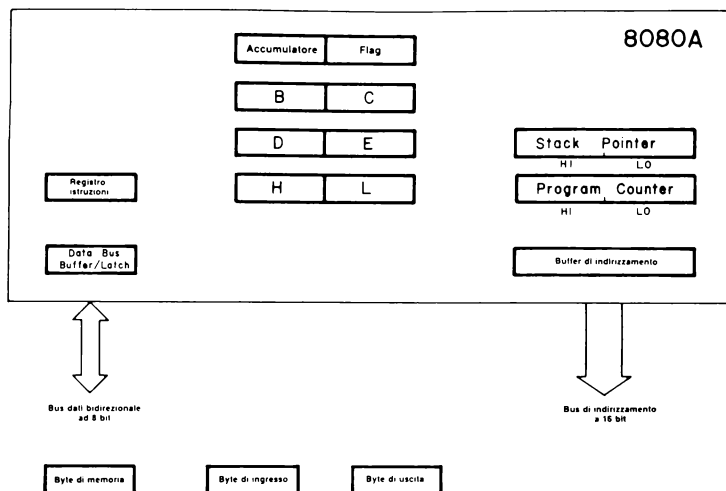



Fig. 3-2. L'architettura interna dei registri del microprocessore 8080. Sono stati omessi i registri temporanei, sui quali non c'è controllo diretto. Il buffer/latch del bus di dati e il buffer degli indirizzi forniscono l'interfaccia fra l'insieme dei circuiti all'interno del chip e i bus esterni.

- Il *registro stack pointer* a 16 bit.
- Il *registro contatore di programma* a 16 bit.

Altri due registri sui quali, in casi particolari, potete avere un controllo, sono i seguenti:

- Il *registro istruzioni* a 8 bit.
- Un *registro flag* a 5 bit nell'unità logico/aritmetica (ALU).

I registri addizionali (vedi Fig. 3-3) che sono necessari per permettere al microprocessore 8080 di eseguire le sue operazioni interne comprendono due registri temporanei a 8 bit usati singolarmente o a coppie, il *registro temporaneo W* e il *registro temporaneo Z;* un accumulatore temporaneo a 8 bit nell'unità logico/aritmetica; e un *registro temporaneo* a 8 bit nell'unità logico/aritmetica. Non potete indirizzare o controllare i contenuti di questi registri temporanei da programma e non saprete quando l'8080 li usa.

Ecco alcune definizioni utili:

Accumulator (Accumulatore) — Il registro e l'insieme di circuiti elettronici digitali associati nell'unità logico/aritmetica (ALU) di un computer, nel quale vengono eseguite le operazioni aritmetiche e logiche.

General-purpose register (Registri universali) — Nel microprocessore 8080, i registri a 8 bit che possono prendere parte alle operazioni aritmetiche e logiche con i contenuti dell'accumulatore.

Fig. 3-3. Schema a blocchi funzionale della CPU dell'8080. Notate il bus di dati interno, che comunica con il bus di dati bidirezionale esterno attraverso un buffer/latch del bus dei dati, situato all'interno del chip 8080.

Instruction code (Codice istruzione)	Un numero binario unico a 8 bit che codifica una operazione che può essere eseguita dal microprocessore 8080.
Instruction decoder (Decodificatore di istruzione)	Un decodificatore interno al microprocessore 8080 che decodifica il codice istruzioni in una serie di azioni che possono essere eseguite dal microprocessore.
Instruction register (Registro istruzione)	Il registro a 8 bit nel microprocessore 8080 che memorizza il codice istruzioni dell'istruzione che viene eseguita.
Program counter (Contatore di programma)	Il registro a 16 bit nel microprocessore 8080 che contiene l'indirizzo di memoria del byte istruzioni seguente che verrà eseguito in un programma.
Stack pointer	Il registro a 16 bit nel microprocessore 8080 che memorizza l'indirizzo di memoria dello stack, che è una regione della memoria che memorizza le informazioni temporanee.

L'*Intellect 8/Mod 80 Microcomputer Development System Reference Manual* fornisce molti paragrafi ben fatti che riassumono i concetti di codice istruzioni, registro istruzioni, e decodificatore

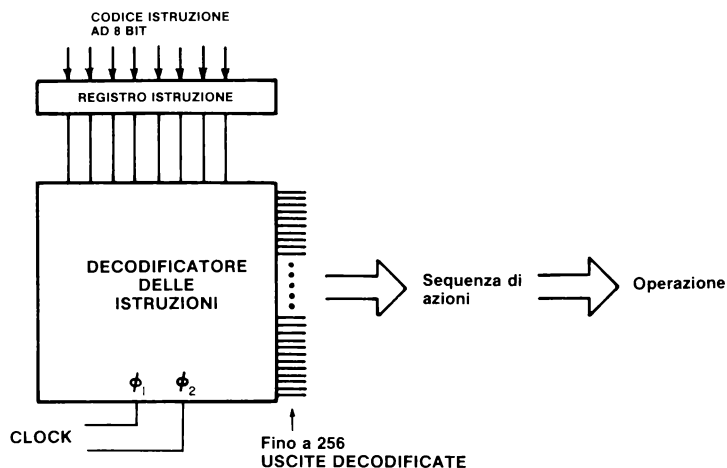



Fig. 3-4. Il codice istruzioni a 8 bit viene dapprima memorizzato nel registro istruzioni, da dove viene decodificato in una serie di azioni temporizzate dal decodificatore di istruzioni all'interno del chip del microprocessore 8080.

di istruzioni. Questi paragrafi vengono citati più avanti (vedi anche Fig. 3-4).

«Ogni computer ha una *lunghezza di parola* che è caratteristica di quella macchina. Nella maggior parte dei sistemi a 8 bit, è più efficace avere a che fare con campi binari a otto bit, e la memoria associata ad un processore di questo tipo è quindi organizzata per memorizzare otto bit in ogni posizione di memoria indirizzabile. I dati e le istruzioni vengono immagazzinati in memoria come numeri binari a otto bit, o come numeri che sono multipli interi di otto bit: 16 bit, 24 bit, e così via. Questo campo caratteristico a otto bit viene chiamato *byte.*

«Ogni operazione che può essere eseguita dal processore viene identificata da un unico numero binario conosciuto come *codice istruzione.* Una parola di otto bit usata come codice istruzioni può distinguere fra 256 azioni alternative, più che adeguate per la maggior parte dei processori.

«Il processore *preleva* dalla memoria un'istruzione in due operazioni distinte. Nella prima, esso trasmette l'indirizzo alla memoria nel suo contatore di programma. Nella seconda, la memoria rimanda il byte indirizzato al processore. La CPU memorizza questo byte istruzioni in un registro noto come *registro istruzione,* e lo usa per dirigere le attività durante il resto del ciclo istruzione.

«Il meccanismo per mezzo del quale il processore traduce un codice istruzione in specifiche azioni di elaborazione richiede un lavoro più elaborato di quello che possiamo fare. Il concetto, comunque, sarà intuitivamente chiaro per un progettista di logica che abbia dell'esperienza. Gli otto bit memorizzati nel registro

istruzioni possono essere decodificati e usati per attivare selettivamente una fra alcune linee di uscita, in questo caso 256. Ogni linea rappresenta un set di attività associate con l'esecuzione di un particolare codice istruzioni. La linea abilitata può essere combinata in coincidenza con gli impulsi di temporizzazione selezionati, per sviluppare elettricamente dei segnali sequenziali che possono essere usati per dare inizio a delle azioni specifiche. Questa traduzione del codice in azione è eseguita dal *decodificatore di istruzione* e dall'insieme di circuiti di controllo ad esso associati».

Quello che ci interessa qui è che il codice istruzione viene tradotto in una sequenza di azioni specifiche.

Il clock a due fasi è di importanza vitale per questo processo. Le azioni possono risultare nel trasferimento dei dati dalla memoria all'accumulatore, o nella somma dei contenuti del registro B con il registro A, o nel complementare l'accumulatore, o in una qualunque delle operazioni specifiche contenute nel set di istruzioni dell'8080. Ciononostante, *ogni operazione specifica eseguita da un'istruzione dell'8080 è il risultato di una o più azioni specifiche causate dal decodificatore di istruzione.*

QUALI TIPI DI OPERAZIONI ESEGUE IL MICROPROCESSORE 8080?

Lo scopo di questo paragrafo non è quello di suddividere il set di istruzione dell'8080 in categorie, ma piuttosto quello di identificare i tipi fondamentali di operazioni che il chip esegue.

- *Trasferire un byte da una locazione (posizione) ad un'altra.*

  Da un registro universale all'altro
  Da un registro universale alla memoria, e viceversa
  Dall'accumulatore alla memoria e viceversa
  Dall'accumulatore ad un registro universale, e viceversa
  Dalla memoria al registro istruzioni
  Dalla memoria al contatore di programma e viceversa
  Dalla memoria allo stack pointer
  Dall'accumulatore ad un latch di uscita
  Da un dispositivo d'ingresso all'accumulatore
  Da un buffer three-state esterno al registro istruzioni
  Dal registro dei flag alla memoria e viceversa
  Da un registro universale allo stack pointer
  Dal contatore di programma allo stack e viceversa
  Dal registro dei flag alla memoria e viceversa
  Dall'accumulatore allo stack e viceversa
  Dal registro dei flag allo stack e viceversa
  Da un dispositivo d'ingresso ad un registro universale
  Da un registro universale ad un dispositivo di uscita
  Da un registro universale al contatore di programma

- *Operazioni aritmetiche e logiche.*

  Eseguire un'operazione di AND sui contenuti di un registro o della memoria con l'accumulatore
  Eseguire un'operazione di OR sui contenuti di un registro o della memoria con l'accumulatore
  Eseguire un'operazione di OR Esclusivo sui contenuti di un registro o della memoria con l'accumulatore
  Confrontare i contenuti di un registro o della memoria con l'accumulatore
  Sommare i contenuti di un registro o della memoria all'accumulatore (con o senza riporto)
  Sottrarre i contenuti di un registro o della memoria dall'accumulatore (con o senza riporto negativo)
  Far ruotare i contenuti dell'accumulatore
  Incrementare i contenuti di un registro universale, di una coppia di registri, dell'accumulatore, della memoria o dello stack pointer
  Decrementare i contenuti di un registro universale, di una coppia di registri, dell'accumulatore, della memoria o dello stack pointer
  Sommare i contenuti di una coppia di registri ai contenuti di una coppia di registri o dello stack pointer
  Eseguire l'aggiustamento decimale dei contenuti dell'accumulatore

- *Operazioni miste.*

  Nessuna operazione
  Alt
  Abilitare il sistema di interruzione
  Disabilitare il sistema d'interruzione
  Complementare l'accumulatore
  Settare il flag di carry
  Complementare il flag di carry

Per la maggior parte del tempo, tutto quello che il microprocessore 8080 fa, è di trasferire un byte da una locazione ad un'altra o di eseguire un'operazione logica o aritmetica. Raramente esso esegue una delle operazioni miste. In altre parole, il chip non calcola solamente, ma sposta byte tutt'intorno.

ISTRUZIONI MNEMONICHE DELL'8080

Vi incoraggiamo ad imparare il più presto possibile i codici mnemonici, in modo che possiate programmare con il linguaggio assembler, leggere altri programmi in linguaggio assembler per l'8080 e migliorare la vostra capacità di capire i set di istruzioni per altri microprocessori. I codici mnemonici delle istruzioni dell'8080 sono elencati in gruppi nel *«Intel 8080 Microcomputer Systems User's Manual»*, che vi raccomandiamo di procurarvi. Qui farete prima un elenco dei codici mnemonici in ordine alfabetico e poi li descriverete dettagliatamente. Questo materiale proviene da due fonti d'informazione.

Intel 8080 Microcomputer Systems User's Manual, Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051, 1975.

The μCOM-8 Software Manual, NEC Microcomputers, Inc., Five Militia Drive, Lexington, Massachussets 02173, 1975.

Siamo grati del permesso accordatoci di usare le fonti suddette.

	Codice istruzione		
Mnemonico	*Ottale*	*Esadecimale*	*Descrizione*
ACI <B2>	316	CE	Somma del byte immediato all'accumulatore (con riporto)
ADC M	216	8E	Somma del contenuto della memoria all'accumulatore (con riporto)
ADC r	21*S*	†	Somma del contenuto del registro all'accumulatore (con riporto)
ADD M	206	86	Somma del contenuto della memoria all'accumulatore
ADD r	20*S*	†	Somma del contenuto del registro all'accumulatore
ADI <B2>	306	C6	Somma del byte immediato all'accumulatore
ANA M	246	A6	AND del contenuto di memoria con l'accumulatore
ANA r	24*S*	†	AND del contenuto del registro con l'accumulatore
ANI <B2>	346	E6	AND del byte immediato con l'accumulatore
CALL <B2> <B3>	315	CD	Richiamo di subroutine in modo incondizionato
CC <B2> <B3>	334	DC	Richiamo di subroutine solo se il flag di riporto è settato
CM <B2> <B3>	374	FC	Richiamo di subroutine solo se il flag di segno e settato
CMA	057	2F	Complemento del contenuto dell'accumulatore
CMC	077	3F	Complemento del flag di carry
CMP M	276	BE	Confronto tra il contenuto della memoria e quello dell'accumulatore
CMP r	27*S*	†	Confronto tra il contenuto del registro e quello dell'accumulatore

CNC <B2> <B3>	324	D4	Richiamo di subroutine solo se il flag di carry è resettato
CNZ <B2> <B3>	304	C4	Richiamo di subroutine solo se il flag di zero è resettato
CP <B2> <B3>	364	F4	Richiamo di subroutine solo se il flag di segno è resettato
CPE <B2> <B3>	354	EC.	Richiamo di subroutine solo se il flag di parità è settato
CPI <B2>	376	FE	Confronto tra il byte immediato ed il contenuto dell'accumulatore dell'accumulatore
CPO <B2> <B3>	344	E4	Richiamo di subroutine solo se il flag di parità è resettato
CZ <B2> <B3>	314	CC	Richiamo di subroutine solo se il flag di zero e settato
DAA	047	27	Aggiustamento decimale del contenuto dell'accumulatore
DAD B	011	09	Somma della coppia di registri B alla coppia dei registri H
DAD D	031	19	Somma della coppia di registri D alla coppia dei registri H
DAD H	051	29	Somma della coppia di registri H alla coppia dei registri H
DAD SP	071	39	Somma della coppia di registri H allo stack pointer
DCR M	065	35	Decremento del contenuto di memoria
DCR r	0*D*5	†	Decremento del contenuto del registro
DCX B	013	0B	Decremento del contenuto della coppia di registri B
DCX D	033	1B	Decremento del contenuto della coppia di registri D
DCX H	053	2B	Decremento del contenuto della coppia di registri H
DCX SP	073	3B	Decremento dello stack pointer
DI	363	F3	Disabilitazione del sistema d'interruzione
EI	373	FB	Abilitazione del sistema d'interruzione
HLT	166	76	Alt incondizionato
IN <B2>	333	DB	Ingresso dati nell'accumulatore
INR M	064	34	Incremento del contenuto di memoria
INR r	0*D*4	†	Incremento del contenuto del registro
INX B	003	03	Incremento dei contenuti della coppia di registri B
INX D	023	13	Incremento dei contenuti della coppia di registri D
INX H	043	23	Incremento dei contenuti della coppia di registri H
INX SP	063	33	Incremento dello stack pointer
JC <B2> <B3>	332	DA	Salto se il flag di carry è settato
JM <B2> <B3>	372	FA	Salto se il flag di segno è settato
JMP <B2> <B3>	303	C3	Salto incondizionato
JNC <B2> <B3>	322	D2	Salto se il flag di carry è resettato
JNZ <B2> <B3>	302	C2	Salto se il flag di zero è resettato
JP <B2> <B3>	362	F2	Salto se il flag di segno è resettato
JPE <B2> <B3>	352	EA	Salto se il flag di parità è settato

JPO <B2> <B3>	342	E2	Salto se il flag di parità è resettato
JZ <B2> <B3>	312	CA	Salto se il flag di zero è settato
LDA <B2> <B3>	072	3A	Caricamento diretto dell'accumulatore con il contenuto della locazione di memoria indirizzata da <B2> <B3>
LDAX B	012	0A	Caricamento indiretto dell'accumulatore con il contenuto della locazione di memoria indirizzata dalla coppia di registri B
LDAX D	032	1A	Caricamento indiretto dell'accumulatore con il contenuto della locazione di memoria indirizzata dalla coppia di registri D
LHLD <B2> <B3>	052	2A	Caricamenti di L ed H con i contenuti delle locazioni di memoria M ed M+1
LXI B <B2> <B3>	001	01	Caricamento immediato della coppia di registri B
LXI D <B2> <B3>	021	11	Caricamento immediato della coppia di registri D
LXI H <B2> <B3>	041	21	Caricamento immediato della coppia di registri H
LXI SP <B2> <B3>	061	31	Caricamento immediato dello stack pointer
MVI M <B2>	066	36	Trasferimento immediato di un byte in una locazione di memoria
MVI r <B2>	0*D*6	†	Trasferimento immediato di un byte in un registro
MOV M,r	16*S*	†	Trasferimento in una locazione di memoria del contenuto di un registro
MOV r,M	1*D*6	†	Trasferimento in un registro del contenuto di una locazione di memoria
MOV r1,r2	1*DS*	†	Trasferimento nel registro 1 del contenuto del registro 2
NOP	000	00	Nessuna operazione
ORA M	266	B6	OR del contenuto di una locazione di memoria con il contenuto dello accumulatore
ORA r	26*S*	†	OR del contenuto di un registro con il contenuto dell'accumulatore
ORI <B2>	366	F6	OR immediato di un byte con il contenuto dell'accumulatore
OUT <B2>	323	D3	Uscita del contenuto dell'accumulatore
PCHL	351	E9	Caricamento del Program Counter con il contenuto della coppia di registri H (salto indiretto)
POP B	301	C1	Prelievo dallo Stack della coppia di registri B
POP D	321	D1	Prelievo dallo Stack della coppia di registri D
POP H	341	E1	Prelievo dallo Stack della coppia di registri H
POP PSW	361	F1	Prelievo dallo Stack del Program Status Word (flag ed accumulatore)
PUSH B	305	C5	Inserimento nello stack della coppia di registri B
PUSH D	325	D5	Inserimento nello stack della coppia di registri D

PUSH H	345	E5	Inserimento nello stack della coppia di registri H
PUSH PSW	365	F5	Inserimento nello stack della Program Status Word (flag ed accumulatore)
RAL	027	17	Rotazione a sinistra del contenuto dell'accumulatore attraverso il carry
RAR	037	1F	Rotazione a destra del contenuto dell'accumulatore attraverso il carry
RC	330	D8	Ritorno se il flag di carry è settato
RET	311	C9	Ritorno incondizionato
RLC	007	07	Rotazione a sinistra del contenuto dell'accumulatore
RM	370	F8	Ritorno se il flag di segno è settato
RNC	320	D0	Ritorno se il flag di carry è resettato
RNZ	300	C0	Ritorno se il flag di zero è resettato
RP	360	F0	Ritorno se il flag di segno è resettato
RPE	350	E8	Ritorno se il flag di parità è settato
RPO	340	E0	Ritorno se il flag di parità è resettato
RRC	017	0F	Rotazione a destra del contenuto dell'accumulatore
RST n	3*N*7	†	Richiamo della subroutine alle locazioni HI=000, LO=On
RZ	310	C8	Ritorno se il flag di zero è settato
SBB M	236	9E	Sottrazione dall'accumulatore del contenuto della locazione di memoria indicata (con riporto negativo)
SBB r	23*S*	†	Sottrazione dall'accumulatore del contenuto del registro indicato (con riporto negativo)
SBI <B2>	336	DE	Sottrazione immediata di un byte dall'accumulatore (con riporto negativo)
SHLD<B2><B3>	042	22	Memorizzazione in M, M+1 del contenuto della coppia di registri H (M=<B2>, <B3>)
SPHL	371	F9	Trasferimento nello stack pointer del contenuto della coppia di registri H
STA <B2> <B3>	062	32	Memorizzazione del contenuto dell'accumulatore direttamente nella locazione di memoria data da <B2>, <B3>
STAX B	002	02	Memorizzazione indiretta del contenuto dell'accumulatore nella locazione di memoria indirizzata dalla coppia di registri B
STAX D	022	12	Memorizzazione indiretta del contenuto dell'accumulatore nella locazione di memoria indirizzata dalla coppia di registri D
STC	067	37	Set del flag di carry
SUB M	226	96	Sottrazione del contenuto di memoria dall'accumulatore
SUB r	22*S*	†	Sottrazione del contenuto del registro dall'accumulatore
SUI <B2>	326	D6	Sottrazione immediata di un byte dall'accumulatore

XCHG	353	EB	Scambio dei contenuti della coppia di registri D e H
XRA M	256	AE	OR-esclusivo memoria-accumulatore
XRA r	25S	+	OR-esclusivo registro-accumulatore
XRI <B2>	356	EE	OR-esclusivo immediato di un byte con l'accumulatore
XTHL	343	E3	Scambio del top dello stack con il contenuto della coppia di registri H

Non tutti i 256 possibili codici istruzione ($2^8=256$) sono utilizzati dall'8080. Alcuni dei codici non usati sono i seguenti:

010	08
020	10
030	18
040	20
050	28
060	30
070	38
313	CB
331	D9
335	DD
355	ED
375	FD

Nota: le istruzioni aventi la notazione «+», non sono facilmente traducibili in codice esadecimale senza informazioni su registri usati, od altro. Questa è una delle ragioni per cui abbiamo scelto i numeri ottali. Procederemo nella descrizione del set di istruzioni dell'8080 in dettaglio.

ELENCO OTTALE/ESADECIMALE DEL SET DI ISTRUZIONI 8080

Oltre all'elenco alfabetico del set di istruzioni dell'8080, relativamente al codice mnemonico, è utile un elenco delle istruzioni in termini di codice ottale. L'elenco è fornito di seguito volendo, potete farne una fotocopia e tenerlo a portata di mano. Noi siamo del parere che esso abbia un particolare valore nello sviluppo di certi programmi in codice macchina.

L'indicazione ——— significa che il byte istruzione non ha effetto sull'8080: non è una istruzione valida.

Codice istruzione			
Ottale	*Esadecimale*	*Mnemonico*	*Descrizione*
000	00	NOP	Nessuna oerazione
001	01	LXI B <B2> <B3>	Carica in modo immediato nella coppia registri B e C
002	02	STAX B	Memorizza A in modo indiretto in M indirizzato da B e C
003	03	INX B	Incrementa di 1 i contenuti della coppia di registri B e C
004	04	INR B	Incrementa di 1 il registro B
005	05	DCR B	Decrementa di 1 il registro B

006	06	MVI B <B2>	Sposta in modo immediato nel registro B
007	07	RLC	Ruota A verso sinistra
010	08	—	——
011	09	DAD B	Somma i contenuti di B,C ad H,L e memorizza in H,L
012	0A	LDAX B	Carica A in modo indiretto da M indirizzato da B e C
013	0B	DCX B	Decrementa di 1 i contenuti della coppia di registri B e C
014	0C	INR C	Incrementa di 1 il registro C
015	0D	DCR C	Decrementa di 1 il registro C
016	0E	MVI C <B2>	Sposta in modo immediato nel registro C
017	0F	RRC	Ruota A verso destra
020	10	—	——
021	11	LXI D <B2> <B3>	Carica in modo immediato nella coppia di registri D e E
022	12	STAX D	Memorizza A in modo indiretto in M indirizzato da D ed E
023	13	INX D	Incrementa di 1 i contenuti della coppia di registri D ed E
024	14	INR D	Incrementa di 1 il registro D
025	15	DCR D	Decrementa di 1 il registro D
026	16	MVI D <B2>	Sposta in modo immediato nel registro D
027	17	RAL	Ruota A a sinistra attraverso il carry
030	18	—	——
031	19	DAD D	Somma i contenuti di D,E ad H,L e memorizza in H,L
032	1A	LDAX D	Carica A in modo indiretto da M indirizzato da D ed E
033	1B	DCX D	Decrementa di 1 i contenuti della coppia di registri D ed E
034	1C	INR E	Incrementa di 1 il registro E
035	1D	DCR E	Decrementa di 1 il registro E
036	1E	MVI E <B2>	Sposta in modo immediato nel registro E
037	1F	RAR	Ruota A verso destra attraverso il carry
040	20	—	——
041	21	LXI H <B2> <B3>	Carica in modo immediato nella coppia di registri H ed L
042	22	SHLD <B2> <B3>	Memorizza L e H in M e M+1, dove M = <B2> <B3>
043	23	INX H	Incrementa di 1 i contenuti della coppia di registri H ed L
044	24	INR H	Incrementa di 1 il registro H
045	25	DCR	Decrementa di 1 il registro H
046	26	MVI H <B2>	Sposta in modo immediato nel registro H
047	27	DAA	Aggiustamento decimale di A
050	28	—	——
051	29	DAD H	Somma i contenuti di H,L ad H,L e memorizza in H,L

052	2A	LHLD <B2> <B3>	Carica L ed H con i contenuti di M e M+1, rispettivamente
053	2B	DCX H	Decrementa di 1 i contenuti della coppia di registri H ed L
054	2C	INR L	Incrementa di 1 il registro L
055	2D	DCR L	Decrementa di 1 il registro L
056	2E	MVI L <B2>	Sposta in modo immediato nel registro L
057	2F	CMA	Complementa A
060	30	—	——
061	31	LXI SP <B2> <B3>	Carica in modo immediato nello stack pointer
062	32	STA <B2> <B3>	Memorizza A in modo diretto in M indirizzato da <B2> <B3>
063	33	INX SP	Incrementa di 1 il registro SP
064	34	INR M	Incrementa di 1 i contenuti di M
065	35	DCR M	Decrementa di 1 i contenuti di M
066	36	MVI M <B2>	Sposta in modo immediato in M indirizzato da H ed L
067	37	STC	Setta il flip-flop di carry al livello logico 1
070	38	—	——
071	39	DAD SP	Somma i contenuti dello stack pointer ad H,L e memorizza in H,L
072	3A	LDA <B2> <B3>	Carica A in modo diretto con i contenuti di M indirizzati da <B2> <B3>
073	3B	DCX SP	Decrementa di 1 il registro SP
074	3C	INR A	Incrementa di 1 il registro A
075	3D	DCR A	Decrementa di 1 il registro A
076	3E	MVI A <B2>	Sposta in modo immediato nel registro A
077	3F	CMC	Complementa il flip-flop di carry
100	40	MOV B,B	Sposta i contenuti del registro B nel registro B
101	41	MOV B,C	Sposta i contenuti del reg. C nel reg. B
102	42	MOV B,D	Sposta i contenuti del reg. D nel reg. B
103	43	MOV B,E	Sposta i contenuti del reg. E nel reg. B
104	44	MOV B,H	Sposta i contenuti nel reg. H nel reg. B
105	45	MOV B,L	Sposta i contenuti nel reg. L nel reg. B
106	46	MOV B,M	Sposta i contenuti di M nel reg. B
107	47	MOV B,A	Sposta i contenuti del reg. A nel reg. B
110	48	MOV C,B	Sposta i contenuti del reg. B nel reg. C
111	49	MOV C,C	Sposta i contenuti del reg. C nel reg. C
112	4A	MOV C,D	Sposta i contenuti del reg. D nel reg. C
113	4B	MOV C,E	Sposta i contenuti del reg. E nel reg. C
114	4C	MOV C,H	Sopsta i contenuti del reg. H nel reg. C
115	4D	MOV C,L	Sposta i contenuti del reg. L nel reg. C
116	4E	MOV C,M	Sposta i contenuti di M nel reg. C
117	4F	MOV C,A	Sposta i contenuti del reg. A nel reg. C
120	50	MOV D,B	Sposta i contenuti del reg. B nel reg. D
121	51	MOV D,C	Sposta i contenuti del reg. C nel reg. D
122	52	MOV D,D	Sposta i contenuti del reg. D nel reg. D
123	53	MOV D,E	Sposta i contenuti del reg. E nel reg. D

124	54	MOV D,H	Sposta i contenuti del reg. H nel reg. D
125	55	MOV D,L	Sposta i contenuti del reg. L nel reg. D
126	56	MOV D,M	Sposta i contenuti di M nel reg. D
127	57	MOV D,A	Sposta i contenuti del reg. A nel reg. D
130	58	MOV E,B	Sposta i contenuti del reg. B nel reg. E
131	59	MOV E,C	Sposta i contenuti del reg. C nel reg. E
132	5A	MOV E,D	Sposta i contenuti del reg. D nel reg. E
133	5B	MOV E,E	Sposta i contenuti del reg. E nel reg. E
134	5C	MOV E,H	Sposta i contenuti del reg. H nel reg. E
135	5D	MOV E,L	Sposta i contenuti del reg. L nel reg. E
136	5E	MOV E,M	Sposta i contenuti di M nel reg. E
137	5F	MOV E,A	Sposta i contenuti del reg. A nel reg. E
140	60	MOV H,B	Sposta i contenuti del reg. B nel reg. H
141	61	MOV H,C	Sposta i contenuti del reg. C nel reg. H
142	62	MOV H,D	Sposta i contenuti del reg. D nel reg. H
143	63	MOV H,E	Sposta i contenuti del reg. E nel reg. H
144	64	MOV H,H	Sposta i contenuti del reg. H nel reg. H
145	65	MOV H,L	Sposta i contenuti del reg. L nel reg. H
146	66	MOV H,M	Sposta i contenuti di M nel reg. H
147	67	MOV H,A	Sposta i contenuti del reg. A nel reg. H
150	68	MOV L,B	Sposta i contenuti del reg. B nel reg. L
151	69	MOV L,C	Sposta i contenuti del reg. C nel reg. L
152	6A	MOV L,D	Sposta i contenuti del reg. D nel reg. L
153	6B	MOV L,E	Sposta i contenuti del reg. B nel reg. L
154	6C	MOV L,H	Sposta i contenuti del reg. H nel reg. L
155	6D	MOV L,L	Sposta i contenuti del reg. L nel reg. L
156	6E	MOV L,M	
157	6F	MOV L,A	Sposta i contenuti del reg. A nel reg. L
160	70	MOV M,B	Sposta i contenuti del reg. B in M
161	71	MOV M,C	Sposta i contenuti del reg. C in M
162	72	MOV M,D	Sposta i contenuti del reg. D in M
163	73	MOV M,E	Sposta i contenuti del reg. E in M
164	74	MOV M,H	Sposta i contenuti del reg. H in M
165	75	MOV M,L	Sposta i contenuti del reg. L in M
166	76	HLT	Alt
167	77	MOV M,A	Sposta i contenuti del reg. A in M
170	78	MOV A,B	Sposta i contenuti del reg. B nel reg. A
171	79	MOV A,C	Sposta i contenuti del reg. C nel reg. A
172	7A	MOV A,D	Sposta i contenuti del reg. D nel reg. A
173	7B	MOV A,E	Sposta i contenuti del reg. E nel reg. A
174	7C	MOV A,H	Sposta i contenuti del reg. H nel reg. A
175	7D	MOV A,L	Sposta i contenuti del reg. L nel reg. A
176	7E	MOV A,M	Sposta i contenuti di M nel reg. A
177	7F	MOV A,A	Sposta i contenuti del reg. A nel reg. A
200	80	ADD B	Somma i contenuti del reg. B al reg. A
201	81	ADD C	Somma i contenuti del reg. C al reg. A
202	82	ADD D	Somma i contenuti del reg. D al reg. A
203	83	ADD E	Somma i contenuti del reg. E al reg. A
204	84	ADD H	Somma i contenuti del reg. H al reg. A
205	85	ADD L	Somma i contenuti del reg. L al reg. A
206	86	ADD M	Somma i contenuti di M al reg. A
207	87	ADD A	Somma i contenuti del reg. A al reg. A

210	88	ADC B	Somma il carry e i cont. del reg. B al reg. A
211	89	ADC C	Somma il carry e i cont. del reg. C al reg. A
212	8A	ADC D	Somma il carry e i cont. del reg. D al reg. A
213	8B	ADC E	Somma il carry e i cont. del reg. E al reg. A
214	8C	ADC H	Somma il carry e i cont. del reg. H al reg. A
215	8D	ADC L	Somma il carry e i cont. del reg. L al reg. A
216	8E	ADC M	Somma il carry e i cont. di M al reg. A
217	8F	ADC A	Somma il carry e i cont. del reg. A al reg. A
220	90	SUB B	Sottrai i cont. del reg. B dal reg. A
221	91	SUB C	Sottrai i cont. del reg. C dal reg. A
222	92	SUB D	Sottrai i cont. del reg. D dal reg. A
223	93	SUB E	Sottrai i cont. del reg. B dal reg. A
224	94	SUB H	Sottrai i cont. del reg. H dal reg. A
225	95	SUB L	Sottrai i cont. del reg. L dal reg. A
226	96	SUB M	Sottrai i cont. di M dal reg. A
227	97	SUB A	Azzera il registro A
230	98	SBB B	Sottrai il carry e i cont. del reg. B da A
231	99	SBB C	Sottrai il carry e i cont. del reg. C da A
232	9A	SBB D	Sottrai il carry e i cont. del reg. D da A
233	9B	SBB E	Sottrai il carry e i cont. del reg. E da A
234	9C	SBB H	Sottrai il carry e i cont. del reg. H da A
235	9D	SBB L	Sottrai il carry e i cont. del reg. L da A
236	9E	SBB M	Sottrai il carry e i cont. di M dal reg. A
237	9F	SBB A	Sottrai il carry e i cont. del reg. A da A
240	A0	ANA B	AND cont. del reg. B con il reg. A
241	A1	ANA C	AND cont. del reg. C con il reg. A
242	A2	ANA D	AND cont. del reg. D con il reg. A
243	A3	ANA E	AND cont. del reg. E con il reg. A
244	A4	ANA H	AND cont. del reg. H con il reg. A
245	A5	ANA L	AND cont. del reg. L con il reg. A
246	A6	ANA M	AND cont. di M con il reg. A
247	A7	ANA A	AND cont. del reg. A con il reg. A
250	A8	XRA B	OR esclusivo cont. del reg. B con il reg. A
251	A9	XRA C	OR esclusivo cont. del reg. C con il reg. A
252	AA	XRA D	OR esclusivo cont. del reg. D con il reg. A
253	AB	XRA E	OR esclusivo cont. del reg. E con il reg. A
254	AC	XRA H	OR esclusivo cont. del reg. H
255	AD	XRA L	OR esclusivo del reg. L con il reg. A
256	AE	XRA M	OR esclusivo cont. di M con il reg. A
257	AF	XRA B	OR esclusivo cont. del reg. A con il reg. A
260	B0	ORA B	OR cont. del reg. B con il reg. A
261	B1	ORA C	OR cont. del reg. C con il reg. A

262	B2	ORA D	OR cont. del reg. D con il reg. A
263	B3	ORA E	OR cont. del reg. E con il reg. A
264	B4	ORA H	OR cont. del reg. H con il reg. A
265	B5	ORA L	OR cont. del reg. L con il reg. A
266	B6	ORA M	OR cont. di M con il reg. A
267	B7	ORA A	OR cont. del reg. A con il reg. A
270	B8	CMP B	Confronta i cont. del reg. B con il reg. A
271	B9	CMP C	Confronta i cont. del reg. C con il reg. A
272	BA	CMP D	Confronta i cont. del reg. D con il reg. A
273	BB	CMP E	Confronta i cont. del reg. E con il reg. A
274	BC	CMP H	Confronta i cont. del reg. H con il reg. A
275	BD	CMP L	Confronta i cont. del reg. L con il reg. A
276	BE	CMP M	Confronta i cont. di M con il reg. A
277	BF	CMP A	Confronta i cont. del reg. A con il reg. A
300	C0	RNZ	Rientra dalla subroutine se il flip-flop di zero = livello logico 0
301	C1	POP B	Estrai dallo stack e memorizza l'indirizzo nella coppia di registri B e C
302	C2	JNZ <B2> <B3>	Salta, se il flip-flop di zero = livello logico 0
303	C3	JMP <B2> <B3>	Salto incondizionato ad M indirizzato da <B2> <B3>
304	C4	CNZ <B2> <B3>	Richiama la subroutine se il flip-flop di zero = livello logico 0
305	C5	PUSH B	Inserisci i contenuti della coppia di registri B e C nello stack
306	C6	ADI <B2>	Somma in modo immediato al reg. A
307	C7	RST 0	Richiama la subroutine all'indirizzo 000_8
310	C8	RZ	Rientra dalla subroutine se il flip-flop di zero = livello logico 1
311	C9	RET	Rientra dalla subroutine
312	CA	JZ <B2> <B3>	Salta se il flip-flop di zero = livello logico 1
313	CB	—	—
314	CC	CZ <B2> <B3>	Richiama la subroutine se il flip-flop di zero = livello logico 1
315	CD	CALL <B2> <B3>	Richiama la subroutine posta a M = <B2> <B3>
316	CE	ACI <B2>	Somma in modo immediato il flip-flop di carry al registro A
317	CF	RST 1	Richiama la subroutine all'indirizzo 010_8
320	D0	RNC	Rientra dalla subroutine se il flip-flop di riporto = livello logico 0
321	D1	POP D	Estrai dallo stack e memorizza l'indirizzo nella coppia di registri D ed E
322	D2	JNC <B2> <B3>	Salta se il flip-flop di carry = livello logico 0

323	D3	OUT <B2>	Poni in uscita verso il dispositivo indirizzato da <B2>
324	D4	CNC <B2> <B3>	Richiama la subroutine se il flip-flop di carry = livello logico 0
325	D5	PUSH D	Inserisci i contenuti della coppia di registri D ed E nello stack
326	D6	SUI <B2>	Sottrai in modo immediato dal reg. A
327	D7	RST 2	Richiama la subroutine all'indirizzo 020_8
330	D8	RC	Rientra dalla subroutine se il flip-flop di carry = livello logico 1
331	D9	—	—
332	DA	JC <B2> <B3>	Salta se il flip-flop di carry = livello logico 1
333	DB	IN <B2>	Poni in ingresso dal dispositivo indirizzato da <B2>
334	DC	CC <B2> <B3>	Richiama la subroutine se il flip-flop di carry = livello logico 1
335	DD	—	—
336	DE	SBI <B2>	Sottrai in modo immediato
337	DF	RST 3	Richiama la subroutine all'indirizzo 030_8
340	E0	RPO	Rientra dalla subroutine se il flip-flop di parità = livello logico 0
341	E1	POP H	Estrai dallo stack e memorizza l'indirizzo nella coppia di reg. H ed L
342	E2	JPO <B2> <B3>	Salta se il flip-flop di parità = livello logico 0
343	E3	XTHL	Scambia la parte superiore dello stack con i contenuti di H ed L
344	E4	CPO <B2> <B3>	Richiama la subroutine se il flip-flop di parità è a livello logico 0
345	E5	PUSH H	Inserisci i contenuti della coppia di registri H ed L nello stack
346	E6	ANI <B2>	AND in modo immediato con i contenuti del registro A
347	E7	RST 4	Richiama la subroutine all'indirizzo 040_8
350	E8	RPE	Rientra dalla subroutine se il flip-flop di parità = livello logico 1
351	E9	PCHL	Salto indiretto ad M indirizzato dalla coppia di registri H ed L
352	EA	JPE <B2> <B3>	Salta se il flip-flop di parità = livello logico 1
353	EB	XCHG	Scambia i contenuti dei registri H, L con i registri D, E
354	EC	CPE <B2> <B3>	Richiama la subroutine se il flip-flop di parità = livello logico 1
355	ED	—	—
356	EE	XRI <B2>	Esegui in modo immediato un'operaz. di OR esclusivo con i cont. del reg. A
357	EF	RST 5	Richiama la subroutine all'indirizzo 050_8
360	F0	RP	Rientra nella subroutine se il flip-flop di segno = livello logico 0
361	F1	POP PSW	Estrai dallo stack memorizza nel registro A e nel PSW
362	F2	JP <B2> <B3>	Salta, se il flip-flop di segno = livello logico 0 (segno positivo)

363	F3	DI	Disabilita l'interruzione
364	F4	CP <B2> <B3>	Richiama la subroutine se il flip-flop di segno = livello logico 0
365	F5	PUSH PSW	Inserisci i contenuti del registro A e i flag sullo stack
366	F6	ORI <B2>	OR in modo immediato con i contenuti del registro A
367	F7	RST 6	Richiama la subroutine all'indirizzo 060_8
370	F3	RM	Rientra dalla subroutine se il flip-flop di segno = livello logico 1
371	F9	SPHL	Trasferisci i contenuti dei registri H,L nello stack pointer
372	FA	JM <B2> <B3>	Salta, se il flip-flop di segno = livello logico 1 (segno meno)
373	FB	EI	Abilita l'interruzione
374	FC	CM <B2> <B3>	Richiama la subroutine se il flip-flop di segno = livello logico 1
375	FD	—	—
376	FE	CPI <B2>	Confronta in modo immediato con i contenuti del registro A
377	FF	RST 7	Richiama la subroutine all'indirizzo 070_8

UN ESEMPIO DI DECODIFICA DELLE ISTRUZIONI

Abbiamo definito in precedenza il codice operativo, conosciuto anche come codice istruzioni, come il codice a 8 bit per l'operazione specifica che il microprocessore 8080 eseguirà. Studiate l'elenco ottale/esadecimale del set di istruzioni dell'8080 dato nelle pagine precedenti. Osserverete che la prima cifra ottale può essere uno 0, un 1, un 2 o un 3 e che le classi specifiche di operazioni corrispondono alla cifra 0, 1, 2 e 3. Possiamo quindi scrivere quanto segue:

Prima Cifra Ottale	*Classe di Operazione*
0	Solo operazioni sui dati, con la possibile eccezione delle istruzioni ottali 067 o 077
1	Solo operazioni di trasferimento dei dati, che hanno tutte in comune il codice mnemonico MOV
2	Solo operazioni aritmetiche e logiche, comprese somma, sottrazione, AND, OR Esclusivo, OR e confronto
3	Operazioni miste, comprese tutte le operazioni condizionate, come salto, chiamata della subroutine e ritorno dalla subroutine

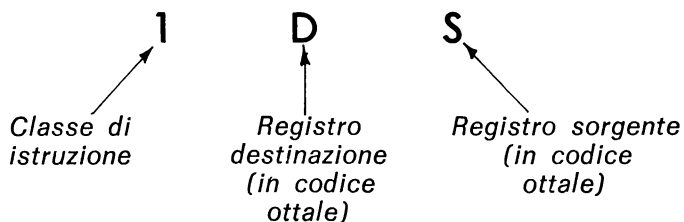
Abbiamo decodificato il set di 244 istruzioni del microprocessore 8080 in quattro classi di operazioni, ognuna delle quali contiene operazioni correlate (ad eccezione forse dell'ultima classe, all'interno della quale le opèrazioni non sono strettamente correlate). Abbiamo fatto per iscritto esattamente la stessa cosa che il decodificatore di istruzioni fa elettronicamente all'interno del microprocessore.

LA DECODIFICA DI UN REGISTRO

Dato che siamo riusciti così bene a decodificare il set di 244 istruzioni in quattro classi distinte, esaminiamo l'elenco del set di istruzioni ottale/esadecimale per altre configurazioni di codice. L'Intel Corporation ci dice che l'accumulatore, i sei registri universali (B, C, D, E, H, L) e la memoria M corrispondono ai seguenti codici binari a 3 bit:

Nome del registro	*Codice a 3 bit*	*Codice ottale*
B	000	0
C	001	1
D	010	2
E	011	3
H	100	4
L	101	5
memoria	110	6
accumulatore (A)	111	7

Prendiamo in esame le 64 istruzioni MOV che iniziano con la cifra ottale 1. Queste istruzioni fanno sì che otto bit di dati vengano trasferiti fra (a) l'accumulatore e un registro universale, (b) l'accumulatore e la memoria, (c) un registro universale e la memoria, o (d) due diversi registri. Le configurazioni relative sono descritte nel paragrafo successivo.

L'istruzione a tre cifre ottali in cui la prima cifra è 1, può essere così rappresentata:

Il registro sorgente è quello che contiene il byte di dati a 8 bit che viene trasferito. Il registro destinazione è quello che riceve il byte di dati a 8 bit che viene trasferito. Così, nell'elenco ottale/esadecimale, «trasferiamo i contenuti del (registro sorgente) nel (registro destinazione)». Il codice mnemonico è:

MOV [registro destinazione], [registro sorgente]

Esaminiamo l'elenco delle istruzioni dell'8080 e verifichiamo le suddette conclusioni. L'istruzione ottale 141 deve significare «trasferire i contenuti del registro C nel registro H», e questo è quello che fa. In queste operazioni, il registro sorgente conterrà ancora i dati trasferiti dato che l'operazione MOV è semplicemente un'operazione di copiatura.

DECODIFICA DELLE OPERAZIONI ARITMETICHE E LOGICHE

La Intel Corporation ci dice anche che le operazioni aritmetiche e logiche corrispondono ai seguenti codici binari a 3 bit:

Operazione aritmetica o logica	*Codice a 3 bit*	*Codice ottale*
Somma	000	0
Somma con riporto	001	1
Sottrazione	010	2
Sottrazione con riporto negativo	011	3
AND	100	4
OR Esclusivo	101	5
OR	110	6
Confronto	111	7

Se esaminiamo le 64 operazioni aritmetiche e logiche che iniziano con la cifra ottale 2, possiamo delineare un'altra configurazione che mostriamo di seguito.

L'istruzione a tre cifre ottali, in cui la prima cifra ottale è 2, può essere così rappresentata:

Nell'elenco ottale/esadecimale, eseguiamo un'operazione aritme-

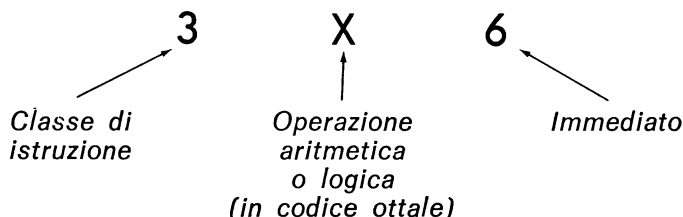
tica o logica usando un registro sorgente (con, da o verso) l'accumulatore. Il codice mnemonico è:

[operazione aritmetica o logica] [registro sorgente]

Esaminiamo l'elenco delle istruzioni dell'8080 e verifichiamo queste addizionali conclusioni. Notiamo di nuovo che stiamo facendo su di un pezzo di carta quello che il decodificatore di istruzioni fa elettronicamente.

Fin qui, abbiamo decodificato 128 delle 244 istruzioni del microprocessore. Stiamo andando bene, per cui continuiamo. Notiamo le otto istruzioni logico/aritmetiche:

306	ADI	<B2>
316	ACI	<B2>
326	SUI	<B2>
336	SBI	<B2>
346	ANI	<B2>
356	XRI	<B2>
366	ORI	<B2>
376	CPI	<B2>

La configurazione è semplicemente la seguente: *l'istruzione a tre cifre ottali, in cui la prima cifra ottale è 3 e l'ultima è 6, si può così rappresentare:*

Secondo l'elenco ottale/esadecimale, eseguiamo un'operazione aritmetica o logica usando il byte a 8 bit immediato (con, da o verso) l'accumulatore. Il codice mnemonico è:

[operazione aritmetica o logica] I

dove i codici mnemonici per le operazioni aritmetiche o logiche sono:

Operazione aritmetica o logica	*Codice mnemonico dell'istruzione immediata*
Somma	ADI
Somma con riporto	ACI
Sottrazione	SUI
Sottrazione con riporto negativo	SBI
AND	ANI
OR Esclusivo	XRI
OR	ORI
Confronto	CPI

DECODIFICA DELLE OPERAZIONI IMMEDIATE

Il termine «*indirizzamento immediato*» viene così definito:

Immediate adressing (Indirizzamento immediato) L'istruzione stessa contiene i dati. Questi dati sono o un solo byte di dati a 8 bit o due byte di dati a 8 bit. Il byte di dati meno significativo è il primo e quello più significativo è il secondo quando ci sono due byte di dati a 8 bit.
Le istruzioni immediate possono essere rappresentate come mostra la Fig. 3-5. Nel paragrafo precedente sono state date otto istruzioni immediate. Di queste istruzioni, le dodici rimanenti sono:

006	MVI B	<B2>	
016	MVI C	<B2>	
026	MVI D	<B2>	
036	MVI E	<B2>	
046	MVI H	<B2>	
056	MVI L	<B2>	
066	MVI M	<B2>	
076	MVI A	<B2>	
001	LXI B	<B2>	<B3>
021	LXI D	<B2>	<B3>
041	LXI H	<B2>	<B3>
061	LXI SP	<B2>	<B3>

(A) Istruzione immediata a due byte.

(B) Istruzione immediata a tre byte.

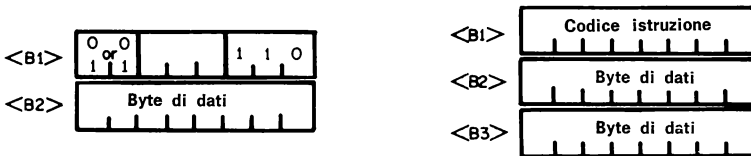



Fig. 3-5. Istruzioni immediate.

Possiamo concludere che *tutte le istruzioni immediate che hanno un solo byte di dati hanno una cifra ottale 0 o 3 come classe di istruzione e 6 come terza cifra ottale.* Il codice mnemonico, MVI [registro destinazione] significa «trasferire il seguente byte di dati a 8 bit nel [registro destinazione]». La posizione di memoria M non è, in effetti, un registro, ma non stiamo a discutere su questi dettagli secondari.

DECODIFICA DELLE OPERAZIONI DI BRANCH (COLLEGAMENTO)

Citiamo ancora una volta l'*Intellec 80/Mod Microcomputer Development System Reference Manual:*

«Le istruzioni che formano un programma vengono memorizzate nella memoria del sistema. Il processore centrale esamina i contenuti della memoria, allo scopo di determinare quale è l'azione appropriata. Ciò significa che il processore deve sapere quale posizione contiene l'istruzione seguente.

«Tutte le locazioni di memoria sono numerate, per distingurle l'una dall'altra nella memoria stessa. Il numero che identifica una posizione di memoria si chiama *indirizzo.*

«Il processore mantiene un contatore che contiene l'indirizzo dell'istruzione seguente del programma. Questo registro si chiama contatore di programma. Il processore aggiorna il contatore di programma aggiungendo "1" al contatore ogni volta che esso preleva dalla memoria un'istruzione, in modo che il contatore di programma è sempre quello attuale.

«Perciò il programmatore memorizza le sue istruzioni in indirizzi numericamente adiacenti, in modo che gli indirizzi più bassi contengano le istruzioni da eseguire prima e gli indirizzi più alti le istruzioni da eseguire dopo. L'unica volta in cui il programmatore può violare questa regola della sequenza è . . .» con un'*istruzione di branch,* come *salto, richiamo di subroutine o ritorno dalla subroutine.*

Possiamo così dare la definizione dei seguenti termini:

Address (Indirizzo)	Il numero binario a 16 bit che identifica una locazione (posizione) di memoria.
Program counter (Contatore di programma)	Il registro che contiene l'indirizzo del byte di istruzioni che deve essere eseguito subito dopo, in un programma.
Branch instruction, branch operation (Istruzione di branch, operazione di branch)	Un'istruzione che fa in modo che un programma salti ad un indirizzo specificato e che venga eseguita l'istruzione a quell'indirizzo. Durante la esecuzione di un istruzione di branch, il processore centrale sostituisce i contenuti del contatore di programma con l'indirizzo specificato.
Jump (Saltare)	1. Fare sì che l'istruzione seguente venga selezionata da una specificata posizione di memoria. 2. Una deviazione dalla normale sequenza dell'esecuzione delle istruzioni.

Call subroutine, call (Chiamare la subroutine, chiamare)	1. Trasferire il controllo ad una subroutine specificata. 2. Un tipo speciale di salto in cui viene richiesto al processore centrale di «ricordare» in modo logico i contenuti del contatore di programma nel momento in cui avviene il salto. Questo permette al processore di riprendere più tardi la esecuzione del programma principale, quando ha concluso l'ultima istruzione della subroutine.
Return from subroutine, return (Ritorno dalla subroutine, ritornare)	Uno speciale tipo di salto in cui il processore riprende l'esecuzione del programma principale al valore del contatore di programma nel momento in cui era avvenuta la chiamata.
Subroutine	1. Nella terminologia dei computer, la parte di un programma che fa sì che un computer porti a termine un'operazione matematica o logica ben definita. 2. Usualmente detta subroutine chiusa. Un piccolo sottoprogramma al quale si può trasferire il controllo dal programma principale, e restituirlo al programma principale alla conclusione della subroutine.
Closed subroutine (Subroutine chiusa)	Subroutine non memorizzata nel flusso principale della routine. Si entra in questa subroutine per mezzo di un'operazione di salto; si provvede a restituire il controllo al programma principale alla fine dell'operazione.
Routine	Set di codici istruzioni posti in un'opportuna sequenza per dirigere il computer nell'esecuzione di un'operazione desiderata o di una sequenza di operazioni. In alternativa, una sottodivisione di un programma che consiste di due o più istruzioni che sono correlate in modo funzionale.
Program (Programma)	Progetto completo per la soluzione di un problema. Più precisamente, la sequenza completa delle istruzioni macchina e delle routine necessarie per risolvere un problema.

Con il microprocessore 8080, le istruzioni di branch, ad eccezione di quelle *incondizionate*, corrispondono al seguente codice a 3 bit per i tre bit meno significativi:

Istruzione di branch	*Codice a 3 bit*	*Codice ottale*
Ritorno	000	0
Salto	010	2
Chiamata	100	4

La classe di istruzione per un'istruzione di branch è sempre 3; questa è la prima cifra ottale dell'istruzione a tre cifre.

Le istruzioni di branch possono essere *condizionate o incondizionate:*

Conditional instruction (Istruzione condizionata)	Nel microprocessore 8080, un'istruzione che è soggetta ad una condizione, ad esempio al livello logico di un flag specifico.

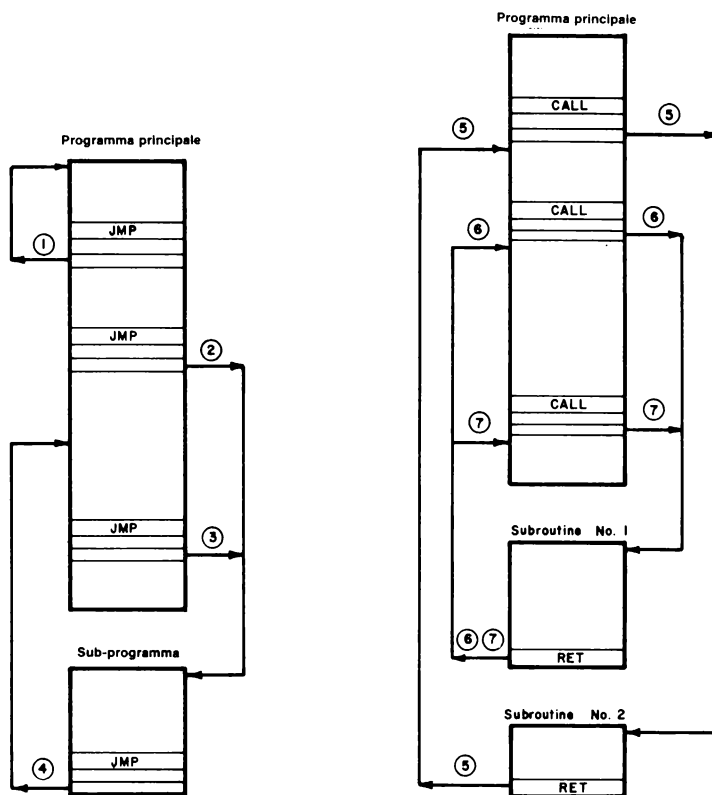



Fig. 3-6. Schema che illustra le differenze fra le istruzioni di salto incondizionato, di richiamo e di ritorno. Un'istruzione di richiamo richiede sempre un'istruzione di ritorno della subroutine.

Unconditional instruction (Istruzione incondizionata) — Nel microprocessore 8080, un'istruzione che non è soggetta a nessuna condizione, quale il livello logico di uno specifico flag.

Parleremo ora delle istruzione di salto, chiamata e ritorno incondizionati, che hanno i seguenti codici istruzioni:

303	<B2> <B3>	salto incondizionato (JMP)
315	<B2> <B3>	chiamata incondizionata (CALL)
311		ritorno incondizionato (RET)

Da notare che l'istruzione di ritorno incondizionato è ad un byte, mentre le altre due sono istruzioni a tre byte.

Con l'aiuto della Fig. 3-6, possiamo vedere le differenze fra le istruzioni di salto, chiamata e ritorno incondizionati.

L'istruzione di salto (1) crea un *loop* nel programma principale nella Fig. 3-6A. Volendo partire all'inizio di questo programma, non ci sarebbe modo di superare questa istruzione di salto. Supponiamo di poterla superare e di procedere alle istruzioni (2) e (3), che saltano nella stessa posizione, un sottoprogramma. Sfortunatamente, da questo sottoprogramma l'istruzione di salto obbliga ad un solo punto del programma principale. Viene così creato un secondo loop come conseguenza delle istruzioni di salto (3) e (4).

Parliamo ora del programma principale e delle due subroutine della Fig. 3-6B. Questo programma procede tranquillamente e non presenta problemi. Quindi:

- L'istruzione di call (5) chiama la subroutine n. 2. Quando questa subroutine è terminata, il controllo di programma ritorna all'*istruzione che segue immediatamente l'istruzione di call* (5).
- L'istruzione di call (6) chiama la subroutine n. 1. Quando questa subroutine è terminata, il controllo del programma torna all'istruzione che segue immediatamente l'istruzione di call 6 .
- Infine, l'istruzione di call (7) chiama la subroutine n. 1. In ogni modo, la seconda volta che la subroutine n. 1 ha termine, il controllo di programma ritorna all'istruzione che segue immediatamente l'istruzione di call (7).

Chiaramente, l'istruzione di call è un'istruzione «intelligente», dato che si ricorda quando è apparsa l'istruzione di call nel programma principale. Al contrario, l'istruzione di salto nel programma di sinistra è un'istruzione non intelligente»; non ricorda da dove è saltata. Le istruzioni di salto si usano raramente per trasferire il controllo ai sottoprogrammi. Al loro posto si usano le istruzioni di call, e vengono aggiunte le istruzioni di ritorno per convertire i sottoprogrammi in subroutine.

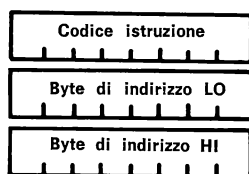
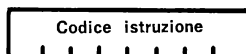
ISTRUZIONI DI SALTO CONDIZIONATO

Ecco come si possono definire le tre istruzioni di salto condizionato:

Conditional jump (Salto condizionato) — Salta se (il flag di condizione) è a (stato di livello logico) alla posizione di memoria indirizzata dai byte B2 e B3. Altrimenti, passare all'istruzione sequenziale successiva.

Conditional call (Chiamata condizionata) — Chiama la subroutine se (il flag di condizione) è a (stato di livello logico) alla posizione di memoria indirizzata dal byte B2 e B3. Altrimenti, passa all'istruzione che segue in sequenza.

Conditional return (Ritorno condizionato) — Ritorna dalla subroutine al programma principale se (il flag di condizione) è a (stato di livello logico). Altrimenti, passa all'istruzione che segue in sequenza.

Le istruzioni di salto e di chiamata condizionati sono istruzioni a tre byte, che possono essere rappresentate da:

mentre le istruzioni di ritorno condizionato sono ad un solo byte:

Una tecnica molto usata per lo sviluppo dei programmi, è quella dei *diagrammi di flusso.*

Flowchart, flow diagram (Diagramma di flusso) — Un diagramma che mostra tutti i passi logici del programma. Un programma viene codificato scrivendo le successive istruzioni che faranno eseguire al computer le operazioni logiche necessarie per risolvere il problema, come sono rappresentate su di un diagramma di flusso.

Flowchart symbol (Simbolo del diagramma di flusso) — Un simbolo usato in un diagramma di flusso per rappresentare i dati, il flusso, gli strumenti, o un'operazione.

Decision (Decisione) — In un computer, il processo di determinazione delle azioni successive sulla base della correlazione esistente fra due gruppi di dati simili fra loro.

Decision symbol (Simbolo decisionale) In un diagramma di flusso, un simbolo usato per contrassegnare una scelta o per attuare il branch nella sequenza di programma di un computer digitale.

Abbiamo divagato su questo argomento perché useremo il simbolo di *«decisione»* che si trova comunemente nei diagrammi di flusso (vedi Fig. 3-7), e le Figg. 3-8, 3-9, 3-10 ci aiuteranno nel trattare l'argomento delle tre istruzioni condizionate. Il simbolo nella Fig. 3-7 rappresenta le seguenti decisioni: se il flag è a livello logico 1, andare all'istruzione alternativa.

L'istruzione JZ della Fig. 3-8 richiede la seguente decisione: se il *flag di zero* è a livello logico 0, il programma esegue l'istruzione che segue in sequenza dopo l'istruzione JZ; se il *flag di zero* è a livello logico 1, il programma salta all'indirizzo di memoria contenuto all'interno del secondo e del terzo byte dell'istruzione. L'istruzione JNZ (vedi Fig. 3-8B) si comporta in modo analogo, ma i flag di condizione sono invertiti.

L'istruzione CZ mostrata in Fig. 3-9A è simile all'istruzione JZ, ma è migliore: se il *flag di zero* è a livello logico 0, il programma esegue l'istruzione che segue in sequenza dopo l'istruzione CZ; se

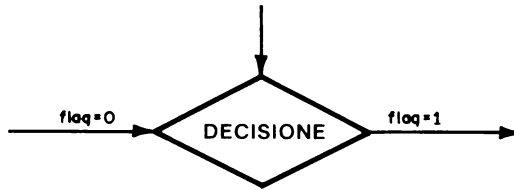



Fig. 3-7. Simbolo della decisione usato nei diagrammi di flusso.

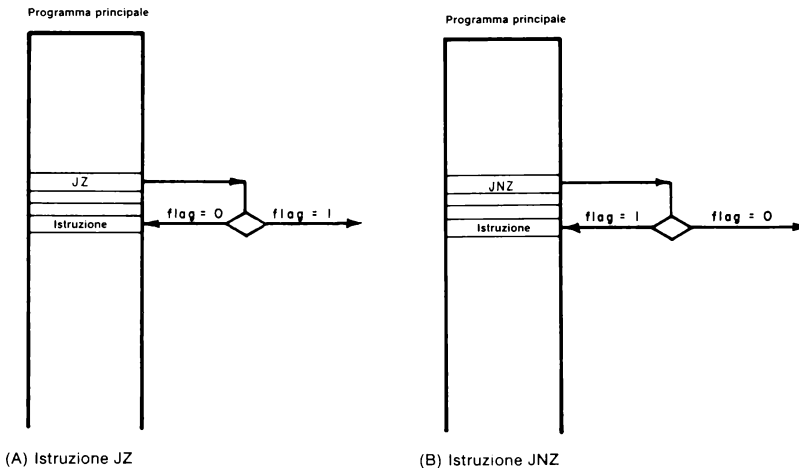



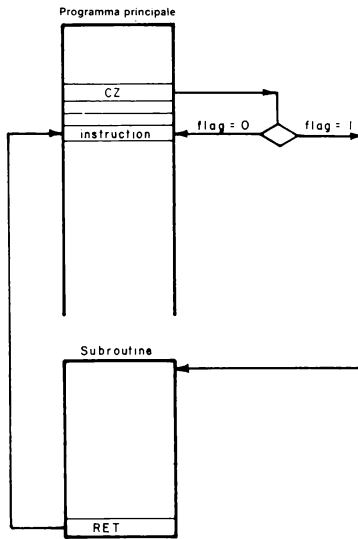
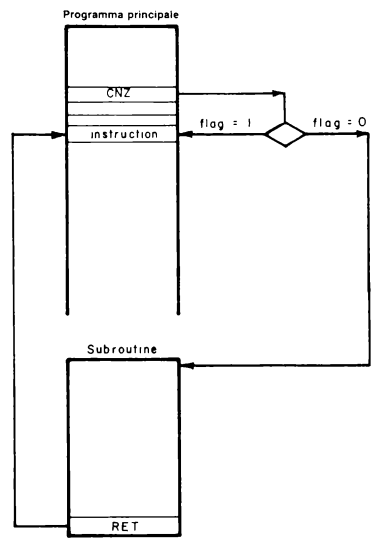
Fig. 3-8. Le istruzioni condizionate JNZ e JZ.

(A) Istruzione CZ

(B) Istruzione CNZ

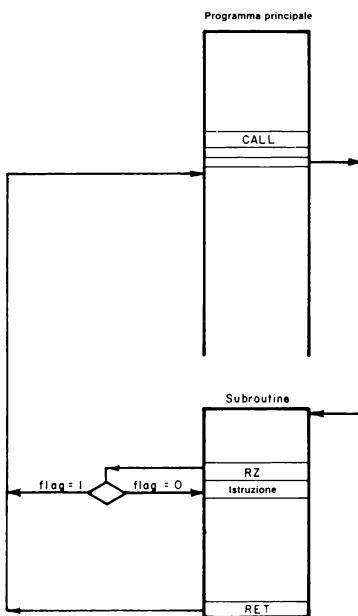
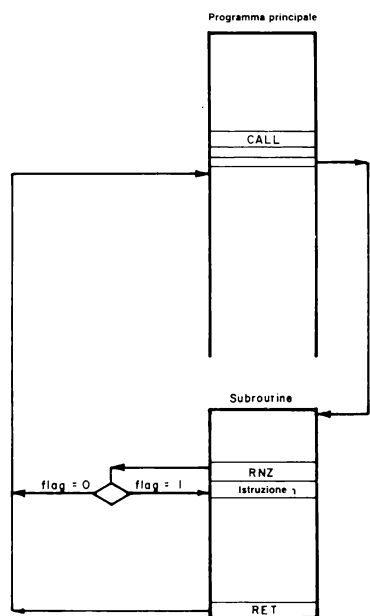
Fig. 3-9. Le istruzioni condizionate CZ e CNZ.

(A) Istruzione RZ

(B) Istruzione RNZ

Fig. 3-10. Diagramma che illustra le istruzioni condizionate RZ e RNZ.

il flag di zero è a livello logico 1, l'indirizzo dell'istruzione seguente viene memorizzato nello *stack*, e il programma salta alla subroutine data dall'indirizzo di memoria contenuto nel secondo e nel terzo byte dell'istruzione.

Con l'istruzione CNZ, le condizioni dei flag sono invertite (Fig. 3-9B). Queste istruzioni sono molto conosciute.

Con l'istruzione RZ mostrata nella Fig. 3-10A, viene presa la seguente decisione: se il *flag di zero* è a livello logico 0, la subroutine esegue l'istruzione che viene in sequenza dopo l'istruzione RZ; se il *flag di zero* è a livello logico 1, viene estratto un indirizzo a 16 bit dallo *stack* e il controllo del programma ritorna al programma principale, all'istruzione che segue immediatamente l'istruzione di chiamata della subroutine, a tre byte. La figura fa rilevare che deve essere eseguita un'istruzione di ritorno nel corso di qualunque passaggio attraverso la subroutine. Se viene saltata l'istruzione RZ, deve essere presente qualche altra istruzione, come RET, per trasferire il controllo di nuovo al programma principale. Con l'istruzione RNZ, le condizioni dei flag sono invertite (Fig. 3-10B).

Nel paragrafo seguente, parleremo più dettagliatamente dei flag di condizione.

DECODIFICAZIONE DEI FLAG DI CONDIZIONE

Un flag è un dispositivo elettronico digitale largamente usato nella programmazione in linguaggio macchina e nell'interfacciamento fra computer e strumenti. Sfortunatamente, non è facile trovare una definizione adatta a questo termine. La migliore sembra essere la seguente:

Flag, condition flag (Flag, flag di condizione) — Un singolo flip-flop che indica che si è verificata una certa condizione nel corso di (a) manipolazioni aritmetiche o logiche in un programma, o (b) trasmissione dati fra due dispositivi elettronici digitali, quali un computer ed uno strumento. Per esempio, un flag indica quando l'accumulatore ha il valore di 000_8. Per fare un altro esempio, un flag può essere un circuito che fornisce un segnale indicante un dispositivo d'ingresso pronto per trasmettere i dati ad un computer.

Quello che conta è che un flag è un flip-flop che fornisce un'informazione ad 1 bit su di una condizione esistente.

Secondo l'Intellec 8/Mod 80 Reference Manual, vi sono cinque flag di condizione associati con l'esecuzione delle istruzioni nel microprocessore 8080. Questi flag sono:

Zero
Carry
Segno

Parità
Carry ausiliario

e sono rappresentati ognuno da un registro ad un bit, o da un flip-flop, contenuti nel microprocessore. Quando leggete le pubblicazioni della Intel, ricordatevi quanto segue:

Quando un flag è «settato», il bit di flag è a livello logico 1;
Quando un flag è «resettato», il bit di flag è a livello logico 0.

Un sinomio di «resettato» è «azzerato». Le istruzioni aritmetiche o logiche eseguite o sui contenuti dell'accumulatore, o sui contenuti di uno dei registri universali, o sui contenuti della memoria, influenzano i flag nei seguenti modi:

Flag di zero — Se il risultato di un'istruzione ha il valore di 000_8, questo flag è settato a livello logico 1; altrimenti, viene resettato a livello logico 0.

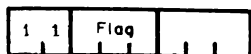
Falg di carry — Se il risultato di un'istruzione produce un riporto (carry) (dall'addizione o dalla rotazione) o un riporto negativo (dalla sottrazione o dal confronto) derivante dal bit più significativo del byte di dati a 8 bit, il flag di carry è settato a livello logico 1; altrimenti, è resettato a livello logico 0.

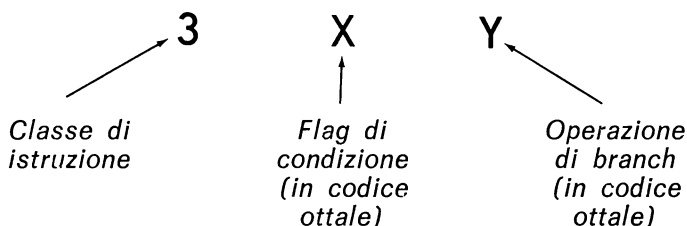
Flag di segno — Se il risultato di un'istruzione produce un livello logico 1 nel bit più significativo del byte di dati a 8 bit, questo flag è settato a livello logico 1 (che indica un risultato negativo); altrimenti, è resettato a livello logico 0 (che indica un risultato positivo).

Flag di parità — Se il risultato di un'istruzione produce un byte di dati a 8 bit, in cui la somma modulo 2 dei bit stessi è a livello logico 0 (che indica parità pari), questo flag è settato a livello logico 1; altrimenti è resettato a livello logico 0 (che indica parità dispari).

Flag di carry ausiliario — Se il risultato di un'istruzione dà luogo ad un riporto dal bit 3 nel bit 4 del byte di dati a 8 bit risultante, il flag di carry ausiliario è settato a livello logico 1; altrimenti, è resettato a livello logico 0. Questo flag è coinvolto da addizioni in singola precisione, sottrazioni, incrementi, decrementi, confronti, ed operazioni logiche, ma viene usato principalmente con le istruzioni di addizione che precedono un'istruzione di DAA (aggiustamento decimale dell'accumulatore)[8].

Quattro dei cinque flag di condizione corrispondono al seguente codice binario a 3 bit, che è contenuto all'interno di tutte le istruzioni di salto, chiamata e ritorno condizionati. Questa parte è mostrata nella rappresentazione per il byte B1 di tutte e tre le istruzioni:

1 1 Flag

Flag di condizione	Codice mnemonico	Stato di livello logico del flag	Codice a 3 bit	Codice ottale
Il risultato non è zero	NZ	0	000	0
Il risultato è zero	Z	1	001	1
Il risultato non ha carry	NC	0	010	2
Il risultato ha carry	C	1	011	3
Il risultato ha parità dispari	PO	0	100	4
Il risultato ha parità pari	PE	1	101	5
Il risultato è positivo	P	0	110	6
Il risultato è negativo (meno)	M	1	111	7

La configurazione per tutte le istruzioni di branch condizionato è quindi:

Le 24 diverse istruzioni condizionate sono riassunte nella Tabella 3-2. (Il flag di carry ausiliario non è un flag di condizione valido; esso ha un uso limitato alla sola istruzione DAA). Potrete capire meglio la Tabella 3-2 facendo riferimento alle Figg. 3-8, 3-9 e 3-10. Sebbene queste tre figure siano state disegnate facendo riferimento alla condizione del flag di zero, dovrebbe essere chiaro che, con i cambiamenti appropriati, esse si applicano altrettanto bene agli altri tre flag di condizione. Alla base di tutte queste condizioni c'è la seguente domanda:

Potrebbe l'istruzione di salto, di chiamata o di ritorno essere ignorata (con il controllo del programma che passa all'istruzione che segue in sequenza) o no (con il controllo del programma che si trasferisce a qualche altra locazione di memoria)?

DECODIFICAZIONE DELLE COPPIE DI REGISTRI

I sei registri universali sono tutti lunghi otto bit. Dato che sono necessari sedici bit per indirizzare qualunque locazione entro le 65.536 possibili posizioni di memoria, conviene avere operazioni che usano coppie di registri invece di registri singoli. In questo modo, si possono trattare parole a 16 bit e si possono indirizzare locazioni di memoria, se necessario, direttamente. Sia lo *stack pointer* che il *contatore di programma* sono registri a 16 bit, per cui sarebbero utili istruzioni che vi permettano di settare i sedici bit in questi registri.

Tabella 3-2. Sommario delle 24 diverse istruzioni condizionate nel set di istruzioni del microprocessore 8080

	Saltare		Chiamare		Ritornare	
Saltare, chiamare o ritornare se il risultato di un'istruzione non è zero	JNZ	302 <B2> <B3>	CNZ	304 <B2> <B3>	RNZ	300
Saltare, chiamare o ritornare se il risultato di un'istruzione è zero	JZ	312 <B2> <B3>	CZ	314 <B2> <B3>	RZ	310
Saltare, chiamare o ritornare se il risultato di un'istruzione non ha carry	JNC	322 <B2> <B3>	CNC	324 <B2> <B3>	RNC	320
Saltare, chiamare o ritornare se il risultato di un'istruzione ha un carry	JC	332 <B2> <B3>	CC	334 <B2> <B3>	RC	330
Saltare, chiamare o ritornare se il risultato di un'istruzione ha parità dispari	JPO	342 <B2> <B3>	CPO	344 <B2> <B3>	RPO	340
Saltare, chiamare o ritornare se il risultato di un'istruzione ha parità pari	JPE	352 <B2> <B3>	CPE	354 <B2> <B3>	RPE	350
Saltare, chiamare o ritornare se il risultato di un'istruzione è positivo	JP	362 <B2> <B3>	CP	364 <B2> <B3>	RP	360
Saltare, chiamare o ritornare se il risultato di un'istruzione è negativo	JM	372 <B2> <B3>	CM	374 <B2> <B3>	RM	370

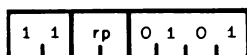
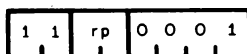
La Intel Corporation ha stabilito quattro coppie di registri che corrispondono ai seguenti codici binari a 2 bit:

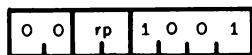
Nome della coppia di registri	*Registri*	*Codice a 2 bit*
B	B e C	00
D	D e E	01
H	H e L	10
PSW	A e flag	11

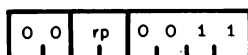
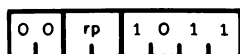
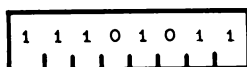
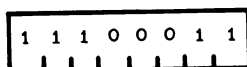
Dato che le coppie di registri si usano per designare gli indirizzi di memoria a 16 bit, è importante identificare un byte d'indirizzo di memoria HI e un byte d'indirizzo di memoria LO. Nelle coppie suddette, i registri B, D, H e i flag sono il byte HI, cioè gli otto bit più significativi nella parola della coppia di registri a 16 bit, e i registri C, E, L e i flag sono i byte LO, cioè gli otto bit meno significativi nella parola della coppia di registri a 16 bit.

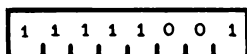
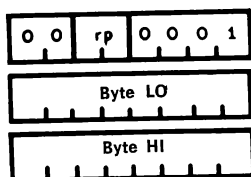
Alcune delle istruzioni delle coppie di registri, come *Push* e *Pop*,

richiedono lunghe spiegazioni. Dato che ne parleremo in un capitolo successivo, nel momento in cui avremo «bisogno di sapere», non scenderemo in dettagli adesso. Seguirà ora un breve riassunto delle operazioni relative alle coppie di registri, e inoltre forniremo la rappresentazione del byte per ogni istruzione delle coppie di registri. Le lettere rp nelle rappresentazioni del byte si riferiscono al codice a 2 bit della coppia di registri, che abbiamo appena dato.

PUSH: Spingere i dati nello stack, istruzione ad un solo byte. I contenuti della coppia di registri specificata vengono conservati in due byte di memoria indicati dallo stack pointer. (Dello stack pointer parleremo più avanti).

POP: Estrarre i dati dallo stack, istruzione ad un solo byte. I contenuti della coppia di registri specificata vengono rimemorizzati dai due byte di memoria indicati dallo stack pointer. (Questa istruzione richiede ulteriori spiegazioni, che daremo più avanti).

DAD: Doppia somma, istruzione ad un byte. Il numero a 16 bit nella coppia di registri specificata viene sommato al numero a 16 bit contenuto nei registri H ed L usando l'aritmetica del complemento a due. Il risultato sostituisce i contenuti dei registri H ed L. Questa è un'istruzione utile per l'indirizzamento indicizzato.

0 0	rp	1 0 0 1

INX: Incrementare la coppia di registri, istruzione ad un solo byte. Il numero a 16 bit contenuto nella coppia di registri specificata viene incrementato di uno.

0 0	rp	0 0 1 1

DCX: Decrementare la coppia di registri, istruzione ad un solo byte. Il numero a 16 bit contenuto nella coppia di registri specificata viene decrementato di uno.

XCHG: Scambiare i registri fra loro, istruzione ad un solo byte. I 16 bit di dati contenuti nei registri H ed L vengono scambiati con i 16 bit di dati dei registri D ed E.

XTHL: Scambiare lo stack, istruzione ad un solo byte. I 16 bit di dati dei registri H ed L vengono scambiati con i 16 bit di dati nei due byte della parte alta dello stack.

SPHL: Caricare lo stack pointer da H ed L, istruzione ad un solo byte. I 16 bit di dati che si trovano nei registri H e L sostituiscono i contenuti dello stack pointer. I contenuti dei registri H e L restano invariati.

1 1 1 1 1 0 0 1

LXI: Caricare la coppia di registri in modo immediato, istruzione a tre byte. Il terzo byte dell'istruzione (gli otto bit più significativi dei dati immediati a 16 bit) viene caricato nel primo registro della coppia specificata, mentre il secondo byte dell'istruzione (gli otto bit meno significativi dei dati immediati a 16 bit) viene caricato nel secondo registro della coppia specificata. *Se lo stack pointer viene specificato come coppia di registri, il secondo byte dell'istruzione sostituisce gli otto bit meno significativi dello stack pointer, mentre il terzo byte dell'istruzione sostituisce gli otto bit più significativi dello stack pointer.*

La Fig. 3-11 riassume molte operazioni delle coppie di registri, diverse fra loro. L'istruzione LXI rp sostituisce i contenuti di una delle quattro paia di registri nel secondo e nel terzo byte dell'istruzione LXI. LXI SP è particolarmente importante per posizionare lo stack pointer immediatamente dopo aver attivato il microcomputer. XCHG, istruzione ad un solo byte, scambia i contenuti della coppia di registri D con i contenuti della coppia di registri H.

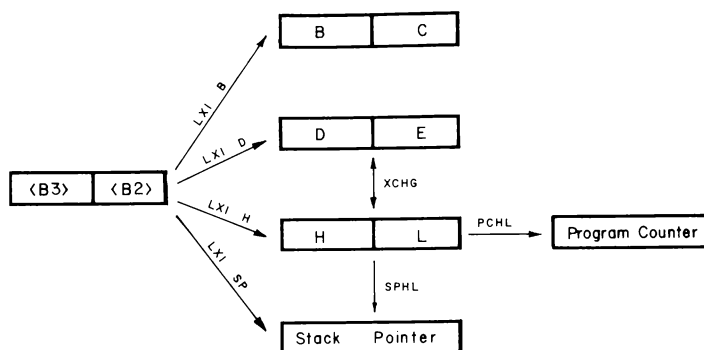



Fig. 3-11. Sommario delle più importanti operazioni delle coppie di registri. Il program counter e lo stack pointer sono entrambi registri a 16 bit.

PCHL sostituisce la parola del contatore di programma a 16 bit con la coppia di registri H. Ecco la definizione di *contatore di programma:*

Program counter (Contatore di programma) Il registro a 16 bit nel microprocessore 8080 che contiene l'indirizzo dell'istruzione successiva o del byte di istruzioni che deve essere eseguito in un programma.

Dovrebbe essere chiaro che l'istruzione PCHL è un'istruzione di salto. Infine, SPHL sostituisce il valore dello stack pointer a 16 bit con il valore della coppia di registri H. E' un altro modo di riposizionare lo stack pointer. Le istruzioni di *pop* e di *push* vengono definite qui di seguito e sono riassunte nella Fig. 3-12.

Pop (Estrarre) Trasferire uno o più byte dallo stack a qualche altro registro o gruppo di registri. Estrarre i dati dallo stack.

Push (Spingere) Sostituire i contenuti di una o più locazioni di memoria in uno stack con i contenuti di un registro o di un gruppo di registri. Inserire i dati nello stack.

Stack L'area di memoria tenuta da parte dal programmatore, in cui vengono memorizzati e poi estratti i dati o gli indirizzi.

Stack pointer L'indirizzo di memoria a 16 bit della parte alta dello stack.

Top of stack (Parte alta dello stack) L'indirizzo di memoria dell'ultimo byte di dati posto nello stack.

L'informazione più comune che risiede in uno stack è l'indirizzo di memoria a cui il programma ritornerà dopo l'esecuzione di una subroutine. Le altre informazioni, come mostra la Fig. 3-12, sono i

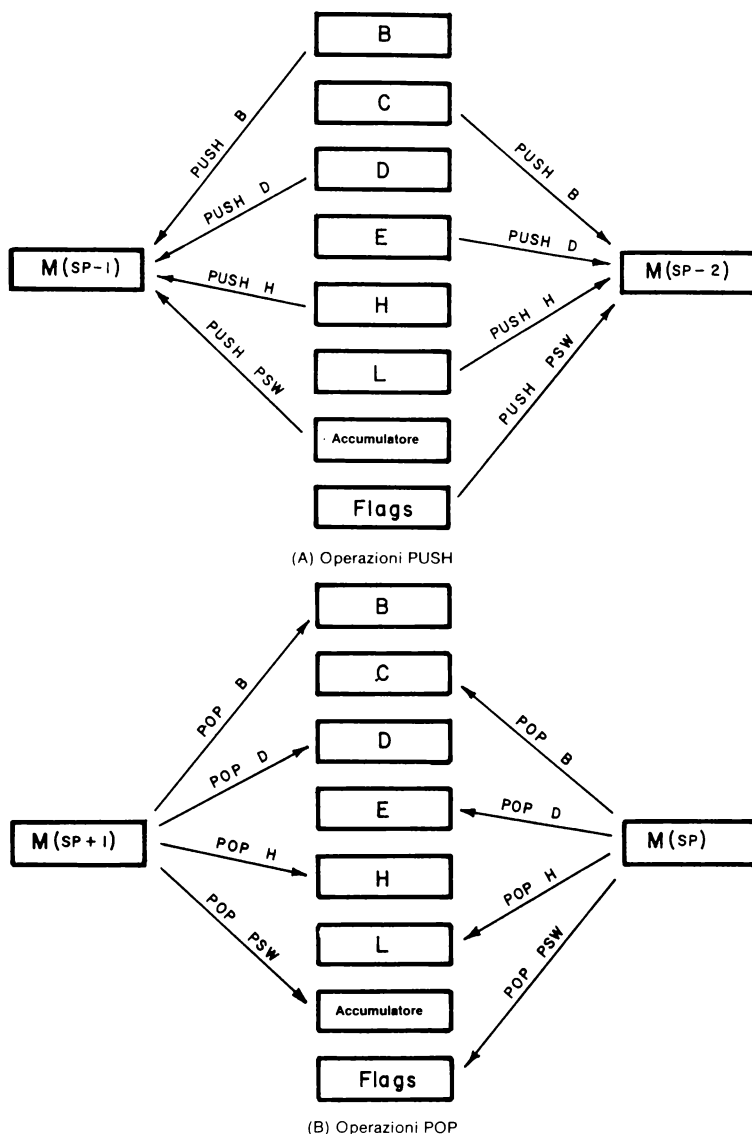



Fig. 3-12. Diagrammi illustranti le operazioni PUSH e POP, ognuna delle quali richiede una coppia di registro e due locazioni di memoria.

contenuti dei sei registri universali, dell'accumulatore e dei flag.

Nella Fig. 3-12B, la posizione di memorizzazione M (SP) sta a significare *i contenuti di quella locazione di memoria il cui indirizzo a 16 bit è contenuto nel registro stack pointer.* I significati di M (SP —2), M (SP —1) e M (SP + 1) dovrebbero essere chiari. Vi sono quattro diverse istruzioni di Push e altrettante di Pop, ognuna delle quali opera con una coppia di registri e due locazioni di memoria all'interno dello stack. Per esempio, nell'istruzione

PUSH B i contenuti del registro B sostituiscono i contenuti della locazione di memoria M (SP —2); dopodiché, lo stack pointer viene decrementato di due per permettere alle istruzioni di Push successive di inserire byte di dati addizionali sullo stack. L'istruzione POP B inverte questo processo. I contenuti di M (SP + 1) sostituiscono i contenuti del registro B e i contenuti di M (SP) sostituiscono i contenuti del registro C; dopodiché, lo stack pointer viene incrementato di due per mettere a disposizione due byte addizionali nella parte alta dello stack che possono essere inseriti in una coppia di registri.

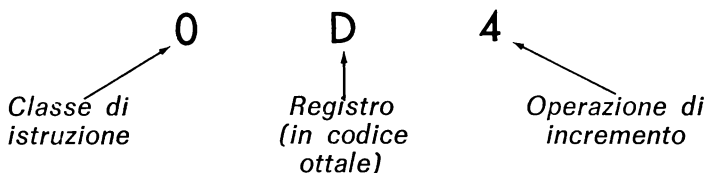
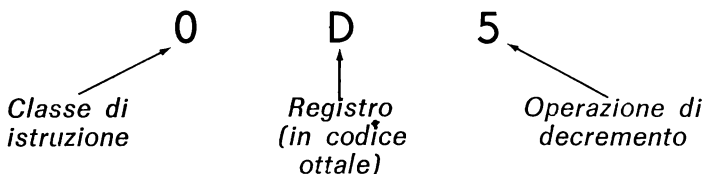
Le operazioni di Push e di Pop verranno trattate ancora nel Capitolo 8.

LA DECODIFICA DELLE OPERAZIONI DI INCREMENTO E DECREMENTO

I termini *«incrementare»* e *«decrementare»* possono essere così definiti:

Decrement (Decrementare)	Aumentare il valore di una parola binaria. Generalmente, aumentare il valore di uno.
Increment (Incrementare)	Diminuire il valore di una parola binaria. Generalmente, diminuire il valore di uno.

La configurazione del codice per le istruzioni di incremento e decremento viene data qui di seguito.

L'istruzione di incremento a tre cifre ottali, in cui la prima cifra ottale è 0 e l'ultima è 4, si può così rappresentare:

L'istruzione di decremento a tre cifre ottali, in cui la prima cifra ottale è 0 e l'ultima è 5, si può così rappresentare:

Le istruzioni di incremento e decremento possono essere definite così:

Incrementare: «Incrementare (il registro) di 1»
Decrementare: «Decrementare (il registro) di 1»

I codici mnemonici sono INR [registro] e DCR [registro].

Abbiamo già parlato di 218 fra le 256 istruzioni del microprocessore. E' una faccenda noiosa, e il pericolo è che stiamo prendendo in considerazione troppe istruzioni diverse senza provarne alcuna. Il nostro interesse principale è stato quello di mostrare in che modo le istruzioni del microprocessore 8080 vengono *decodificate* dal decodificatore di istruzioni. Dovrebbe essere evidente la presenza di un set consistente di regole di codifica dietro tutto il set di istruzioni dell'8080. I registri, le coppie di registri, le operazioni logiche, le istruzioni condizionate, e simili, hanno tutte configurazioni di codice specifiche nel set di istruzioni dell'8080.

METODI D'INDIRIZZAMENTO DEI DATI E DELLA MEMORIA

Dato che tutto il programma e tutti i dati del computer si trovano nella memoria, dovrebbe essere chiaro che l'indirizzamento di locazioni di memoria individuali allo scopo, per esempio, di acquisire byte di dati, è una parte importante di qualunque programma. Tale indirizzamento può essere eseguito in vari modi, fra i quali:

Direct addressing (Indirizzamento diretto) — Viene acquisito un byte di dati a 8 bit per mezzo di un'istruzione a tre byte che contiene l'indirizzo di memoria a 16 bit nel quale il byte di dati è posizionato. Il byte B2 contiene l'indirizzo di memoria LO e il byte B3 contiene l'indirizzo di memoria HI.

Indirect addressing (Indirizzamento indiretto) — Viene acquisito un byte di dati a 8 bit per mezzo di un'istruzione ad un byte che usa una coppia di registri, di solito H ed L, per generare l'indirizzo di memoria a 16 bit. L'indirizzo di memoria HI viene memorizzato in H e l'indirizzo di memoria LO in L. Spesso denotiamo con la lettera M la locazione di memoria indirizzata dai registri H ed L.

Immediate addressing (Indirizzamento immediato) — Viene acquisito un byte a 8 bit per mezzo di una istruzione a due byte che contiene il byte di dati come byte B2.

Stack pointer addressing (Indirizzamento dello stack pointer) — Vengono acquisiti due byte di dati a 8 bit per mezzo di un'istruzione ad un byte che trasferisce i dati da un'area di memoria chiamata stack ad una coppia di registri (o, se si desidera, al registro contatore di programma). Un registro stack pointer fornisce la posizione dello stack, e viene incrementato automaticamente di due dopo ogni istruzione di POP. I dati nello stack sono stati inseriti originariamente da un'operazione di PUSH.

Nelle istruzioni del microprocessore comprese nel «gruppo trasferimento dati», la locazione di memoria M si riferisce a quella posizione indirizzata dai contenuti della coppia di registri H ed L.

ISTRUZIONI RELATIVE ALL'ACCUMULATORE

I diversi tipi di istruzioni dell'accumulatore, descritte in questo capitolo, si possono classificare in questo modo:

- Caricare un byte di dati a 8 bit nell'accumulatore, o dalla memoria, o da un altro registro, o dal secondo byte di un'istruzione immediata.
- Trasferire un byte di dati a 8 bit dall'accumulatore in una locazione di memoria o in un altro registro.
- Caricare un byte di dati a 8 bit nell'accumulatore da un dispositivo d'ingresso.
- Trasferire un byte di dati a 8 bit dall'accumulatore in un dispositivo di uscita.
- Far ruotare i contenuti dell'accumulatore.
- Complementare l'accumulatore.
- Incrementare o decrementare i contenuti dell'accumulatore.
- Spingere i contenuti dell'accumulatore nello stack, o estrarre dallo stack e sostituire i contenuti dell'accumulatore.
- Sommare, sottrarre, confrontare, eseguire operazioni di AND, OR, OR Esclusivo sui contenuti di un registro, di un byte immediato, o di una locazione di memoria, con i contenuti dell'accumulatore e memorizzare il risultato nell'accumulatore.
- Azzerare l'accumulatore.

Di seguito vengono esaminate importanti istruzioni che meritano ulteriori commenti.

Far Ruotare i Contenuti dell'Accumulatore

La Fig. 3-13 fornisce rappresentazioni schematiche per le quattro diverse istruzioni di rotazione.

007 RLC Far ruotare i contenuti dell'accumulatore di una posizione verso sinistra. Il bit meno significativo e il flag di carry sono settati entrambi al valore spostato dalla posizione del bit più significativo del byte dell'accumulatore a 8 bit.

017 RRC Far ruotare i contenuti dell'accumulatore di una posizione verso destra. Il bit più significativo e il flag di carry sono entrambi settati al valore spostato dalla posizione del bit meno significativo del byte a 8 bit dell'accumulatore.

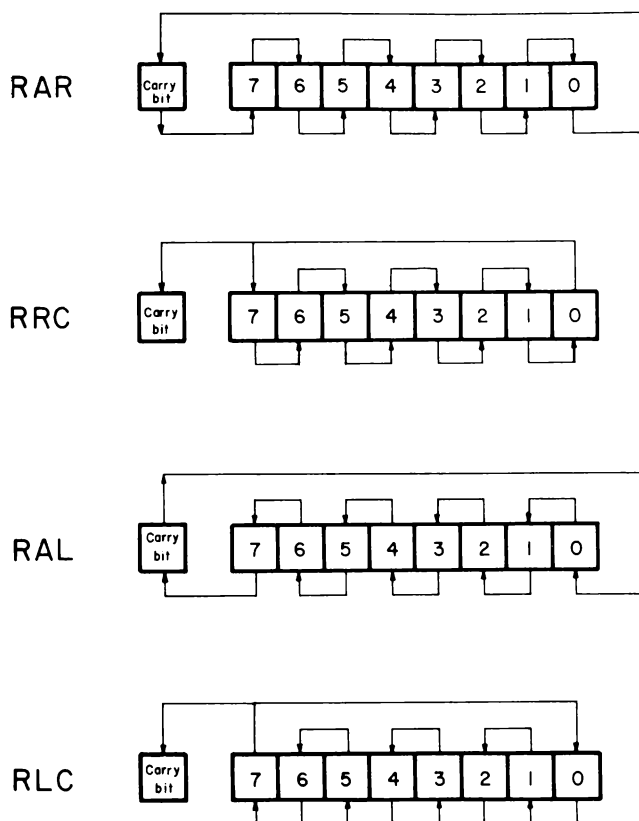



Fig. 3-13. Rappresentazioni delle quattro diverse istruzioni di rotazione nel set di istruzione dell'8080.

027 RAL Far ruotare i contenuti dell'accumulatore di una posizione verso sinistra attraverso il flag di carry. Il bit meno significativo è settato uguale al bit di carry e il flag di carry è settato al valore spostato dal bit più significativo del byte ad 8 bit dell'accumulatore.

037 RAR Far ruotare i contenuti dell'accumulatore di una posizione verso destra attraverso il flag di carry. Il bit più significativo è settato uguale al bit di carry e il flag di carry è settato al valore spostato dal bit meno significativo del byte a 8 bit dell'accumulatore.

Il bit di carry e il byte a 8 bit dell'accumulatore possono essere visti come una parola binaria a 9 bit, in cui il nono bit diventa di livello logico 1 in qualunque momento la somma di due parole binarie a 8 bit produca un riporto dal bit più significativo (bit 7). Le

istruzioni di rotazione sono utili nelle seguenti situazioni:

- Durante la moltiplicazione di una coppia di byte di dati binari a 8 bit. Per moltiplicare per 2 un byte di dati, spostare semplicemente tutto il byte di una posizione verso sinistra.
- Durante la divisione di una coppia di byte di dati binari a 8 bit. Per dividere per 2 un byte di dati, spostare semplicemente tutto il byte di una posizione verso destra.
- Durante il test di bit di flag esterni inseriti nell'accumulatore con l'aiuto di un'istruzione IN del microprocessore. Ogni flag esterno può essere sottoposto a test a turno, facendo ruotare l'accumulatore attraverso il bit di carry ed eseguendo il test dello stato del livello logico del bit di carry.

La rappresentazione schematica per l'istruzione RAR si può spiegare in questo modo:

> Il bit di carry viene spostato al bit 7 del byte dell'accumulatore. Simultaneamente, il bit 7 viene spostato al bit 6, il bit 6 al bit 5, il bit 5 al bit 4, il bit 4 al bit 3, il bit 3 al bit 2, il bit 2 al bit 1, il bit 1 al bit 0, e il bit 0 al bit di carry.

Le istruzioni di rotazione che restano si possono spiegare in modo analogo. Le frecce nelle rappresentazioni indicano le destinazioni di ognuno dei bit mostrati.

Aggiustamento Decimale dei Contenuti dell'Accumulatore

Lo scopo dell'istruzione DAA, **047**, è quello di convertire il risultato dell'addizione di due numeri bcd attraverso l'addizione binaria diretta in due parole bcd impaccate, ognuna delle quali contiene quattro bit. Questa istruzione può, all'occasione, risultare piuttosto confusa. Citiamo direttamente l'*Intel 8080 Assembly Language Programming Manual* (1974):

047 DAA «Il numero esadecimale a otto bit nell'accumulatore viene accomodato in modo da formare due cifre bcd a quattro bit con il seguente processo in due fasi:

> (1) Se i quattro bit meno significativi dell'accumulatore rappresentano un numero maggiore di 9, o se il bit di riporto ausiliario è a livello logico 1, i quattro bit più significativi dell'accumulatore vengono incrementati di sei. Altrimenti, non si verifica nessun incremento.
>
> (2) Se i quattro bit più significativi dell'accumulatore rappresentano ora un numero maggiore di 9, o se il bit di carry è a livello logico 1, i quattro bit più significativi dell'accumulatore vengono incre-

mentati di sei. Altrimenti, non si verifica nessun incremento.

Se c'è un riporto dai quattro bit meno significativi durante la fase (1), il bit di carry ausiliario è settato a livello logico 1; altrimenti è resettato a livello logico 0. Analogamente, se c'è un riporto dai quattro bit più significativi durante la fase (2), il bit normale di carry è settato a livello logico 1; altrimenti, non viene coinvolto.

NOTA: Questa istruzione si usa quando si sommano i numeri decimali. E' l'unica istruzione la cui operazione interessa il bit di carry ausiliario.

Bit di condizione coinvolti: Zero, Segno, Parità, Carry, Carry Ausiliario».

Quando usate questa istruzione molto insolita, vorremmo esortarvi a tener presente la seguente regola:

Usate l'istruzione DAA (Aggiustamento decimale dell'accumulatore) solo dopo un'istruzione ADD, ADC, o ADI.

Perché? Perché solo dopo una di queste istruzioni potete essere sicuri che gli stati dei bit di carry e di carry ausiliario siano giusti. Volendo, potete aggiungere zero ad un byte a 8 bit che volete aggiustare secondo il sistema decimale. Potete farlo con tre byte istruzioni consecutivi :

306	ADI	Sommare il byte seguente ai contenuti dell'accumulatore
000	000	Zero
047	047	Aggiustamento decimale dei contenuti dell'accumulatore

Questi tre byte istruzioni vi permetteranno di aggiustare secondo il sistema decimale i contenuti dell'accumulatore dopo le seguenti istruzioni: IN <B2> e INR A. Gli autori hanno tentato con DCR A e SUI <B2>, ma non hanno riscontrato nessuna utilità. La conclusione è che dovreste stare molto attenti con l'istruzione DAA. Potreste tentare di eseguire vari programmi che la incorporino e dimostrino a voi stessi come tragga in inganno.

Azzerare l'Accumulatore

Vi sono due istruzioni che vi permetteranno di azzerare l'accumulatore, cioè di settare i contenuti del registro accumulatore a 00000000_2:

227	SUB A	Sottrarre i contenuti dell'accumulatore dai contenuti dell'accumulatore

257 XRA A Eseguire un'operazione di OR Esclusivo sui contenuti dell'accumulatore con i contenuti dell'accumulatore.

Entrambe le istruzioni impiegano solo 2 μs.

SUDDIVISIONE IN GRUPPI DELLE ISTRUZIONI DELL'8080

Per concludere la nostra discussione sulla decodifica delle istruzioni, vi presentiamo un comodo riferimento per il set dell'8080, in cui le 256 istruzioni sono suddivise nei 5 gruppi suggeriti dalla Intel Corporation.

1. Gruppo di Trasferimento Dati
2. Gruppo Aritmetico
3. Gruppo Logico
4. Gruppo di Branch
5. Gruppo di Stack, I/O e Controllo Macchina.

In ciascun gruppo, viene presentato il codice ottale, il codice numerico (Intel) e la descrizione delle operazioni attuate dalle singole istruzioni.

Gruppo di Trasferimento Dati

100	MOV B,B	Trasferimento dei contenuti del registro B nel registro B
101	MOV B,C	Trasferimento dei contenuti del registro C nel registro B
102	MOV B,D	Trasferimento dei contenuti del registro D nel registro B
103	MOV B,E	Trasferimento dei contenuti del registro E nel registro B
104	MOV B,H	Trasferimento dei contenuti del registro H nel registro B
105	MOV B,L	Trasferimento dei contenuti del registro L nel registro B
106	MOV B,M	Trasferimento dei contenuti della locazione di memoria M nel registro B
107	MOV B,A	Trasferimento dei contenuti dell'accumulatore nel registro B
110	MOV C,B	Trasferimento dei contenuti del registro B nel registro C
111	MOV C,C	Trasferimento dei contenuti del registro C nel registro C
112	MOV C,D	Trasferimento dei contenuti del registro D nel registro C
113	MOV C,E	Trasferimento dei contenuti del registro E nel registro C
114	MOV C,H	Trasferimento dei contenuti del registro H nel registro C
115	MOV C,L	Trasferimento dei contenuti del registro L nel registro C
116	MOV C,M	Trasferimento dei contenuti della locazione di memoria M nel registro C
117	MOV C,A	Trasferimento dei contenuti dell'accumulatore nel registro C
120	MOV D,B	Trasferimento dei contenuti del registro B nel registro D
121	MOV D,C	Trasferimento dei contenuti del registro C nel registro D
122	MOV D,D	Trasferimento dei contenuti del registro D nel registro D
123	MOV D,E	Trasferimento dei contenuti del registro E nel registro D

124	MOV D,H	Trasferimento dei contenuti del registro H nel registro D
125	MOV D,L	Trasferimento dei contenuti del registro L nel registro D
126	MOV D,M	Trasferimento dei contenuti della locazione di memoria M nel registro D
127	MOV D,A	Trasferimento dei contenuti dell'accumulatore nel registro D
130	MOV E,B	Trasferimento dei contenuti del registro B nel registro E
131	MOV E,C	Trasferimento dei contenuti del registro C nel registro E
132	MOV E,D	Trasferimento dei contenuti del registro D nel registro E
133	MOV E,E	Trasferimento dei contenuti del registro E nel registro E
134	MOV E,H	Trasferimento dei contenuti del registro H nel registro E
135	MOV E,L	Trasferimento dei contenuti del registro L nel registro E
136	MOV E,M	Trasferimento dei contenuti della locazione di memoria M nel registro E
137	MOV E,A	Trasferimento dei contenuti dell'accumulatore nel registro E
140	MOV H,B	Trasferimento dei contenuti del registro B nel registro H
141	MOV H,C	Trasferimento dei contenuti del registro C nel registro H
142	MOV H,D	Trasferimento dei contenuti del registro D nel registro H
143	MOV H,E	Trasferimento dei contenuti del registro E nel registro H
144	MOV H,H	Trasferimento dei contenuti del registro H nel registro H
145	MOV H,L	Trasferimento dei contenuti del registro L nel registro H
146	MOV H,M	Trasferimento dei contenuti della locazione di memoria M nel registro H
147	MOV H,A	Trasferimento dei contenuti dell'accumulatore nel registro H
150	MOV L,B	Trasferimento dei contenuti del registro B nel registro L
151	MOV L,C	Trasferimento dei contenuti del registro C nel registro L
152	MOV L,D	Trasferimento dei contenuti del registro D nel registro L
153	MOV L,E	Trasferimento dei contenuti del registro E nel registro L
154	MOV L,H	Trasferimento dei contenuti del registro H nel registro L
155	MOV L,L	Trasferimento dei contenuti del registro L nel registro L
156	MOV L,M	Trasferimento dei contenuti della locazione di memoria M nel registro L
157	MOV L,A	Trasferimento dei contenuti dell'accumulatore nel registro L
160	MOV M,B	Trasferimento dei contenuti del registro B nella locazione di memoria M

161	MOV M,C	Trasferimento dei contenuti del registro C nella locazione di memoria M
162	MOV M,D	Trasferimento dei contenuti del registro D nella locazione di memoria M
163	MOV M,E	Trasferimento dei contenuti del registro E nella locazione di memoria M
164	MOV M,H	Trasferimento dei contenuti del registro H nella locazione di memoria M
165	MOV M,L	Trasferimento dei contenuti del registro L nella locazione di memoria M
166	HLT	Alt
167	MOV M,A	Trasferimento dei contenuti dell'accumulatore nella locazione di memoria M
170	MOV A,B	Trasferimento dei contenuti del registro B nell'accumulatore
171	MOV A,C	Trasferimento dei contenuti del registro C nell'accumulatore
172	MOV A,D	Trasferimento dei contenuti del registro D nell'accumulatore
173	MOV A,E	Trasferimento dei contenuti del registro E nell'accumulatore
174	MOV A,H	Trasferimento dei contenuti del registro H nell'accumulatore
175	MOV A,L	Trasferimento dei contenuti del registro L nell'accumulatore
176	MOV A,M	Trasferimento dei contenuti della locazione di memoria M nell'accumulatore
177	MOV A,A	Trasferimento dei contenuti dell'accumulatore nell'accumulatore
006	MVI B <B2>	Trasferimento immediato del byte B2 nel registro B
016	MVI C <B2>	Trasferimento immediato del byte B2 nel registro C
026	MVI D <B2>	Trasferimento immediato del byte B2 nel registro D
036	MVI E <B2>	Trasferimento immediato del byte B2 nel registro E
046	MVI H <B2>	Trasferimento immediato del byte B2 nel registro H
056	MVI L <B2>	Trasferimento immediato del byte B2 nel registro L
066	MVI M <B2>	Trasferimento immediato del byte B2 nella locazione di memoria M
076	MVI A <B2>	Trasferimento immediato del byte B2 nell'accumulatore
001	LXI B <B2> <B3>	Caricamento immediato dei due byte B2 e B3 nella coppia di registri B
021	LXI D <B2> <B3>	Caricamento immediato dei due byte B2 e B3 nella coppia di registri D
041	LXI H <B2> <B3>	Caricamento immediato dei due byte B2 e B3 nella coppia di registri H
061	LXI SP <B2> <B3>	Caricamento immediato dei due byte B2 e B3 nella coppia di registri SP
002	STAX B	Memorizzazione indiretta del contenuto dell'accumulatore nella locazione di memoria indirizzata dalla coppia di registri B
012	LDAX B	Caricamento indiretto nell'accumulatore dalla locazione di memoria indirizzata dalla coppia di registri B

022	STAX D	Memorizzazione indiretta del contenuto dell'accumulatore nella locazione di memoria indirizzata dalla coppia di registri D
032	LDAX D	Caricamento indiretto nell'accumulatore dalla locazione di memoria indirizzata dalla coppia di registri D
042	SHLD <B2> <B3>	Memorizzazione diretta di L nella locazione di memoria M indirizzata dai due byte B2, B3; memorizzazione diretta di H nella successiva locazione di memoria
052	LHLD <B2> <B3>	Caricamento diretto di L dalla locazione di memoria M indirizzata dai due byte B2, B3; caricamento diretto di H dalla successiva locazione di memoria.
062	STA <B2> <B3>	Memorizzazione diretta dell'accumulatore nella locazione di memoria indirizzata dai due byte B2 e B3
072	LDA <B2> <B3>	Caricamento diretto dell'accumulatore dalla locazione di memoria indirizzata dai due byte B2 e B3
353	XCHG	Scambio tra i contenuti dei registri H ed L con quelli di D ed E
371	SPHL	Trasferimento dei contenuti dei registri H ed L nel registro Stack Pointer.

Il precedente gruppo di trasferimento dati realizza il trasferimento da e verso registri e memoria. *I flag di condizione non sono interessati da nessuna istruzione di questo gruppo.*

Gruppo Aritmetico

Questo gruppo di istruzioni realizza operazioni aritmetiche sui dati presenti nei registri e in memoria. *Con alcune eccezioni, tutte le istruzioni in questo gruppo modificano i flag di zero, segno, parità e carry, in base alle regole standard.*

Le sole eccezioni sono INR e DCR (carry non modificato), INX e DCX (nessun flag modificato) DAD (modificato solo il flag di carry). Tutte le operazioni di sottrazione sono realizzate tramite la logica del complemento a due e settano il flag di carry ad 1 logico per indicare un «borrow» oppure lo resettano per indicare «no borrow».

200	ADD B	Somma dei contenuti del registro B con i contenuti dell'accumulatore
201	ADD C	Somma dei contenuti del registro C con i contenuti dell'accumulatore
202	ADD D	Somma dei contenuti del registro D con i contenuti dell'accumulatore
203	ADD E	Somma dei contenuti del registro E con i contenuti dell'accumulatore
204	ADD H	Somma dei contenuti del registro H con i contenuti dell'accumulatore
205	ADD L	Somma dei contenuti del registro L con i contenuti dell'accumulatore
206	ADD M	Somma dei contenuti della locazione di memoria M con i contenuti dell'accumulatore
207	ADD A	Somma dei contenuti dell'accumulatore con i contenuti dell'accumulatore

210	ADC B	Somma del bit di carry e dei contenuti del registro B con i contenuti dell'accumulatore
211	ADC C	Somma del bit di carry e dei contenuti del registro C con i contenuti dell'accumulatore
212	ADC D	Somma del bit di carry e dei contenuti del registro D con i contenuti dell'accumulatore
213	ADC E	Somma del bit di carry e dei contenuti del registro E con i contenuti dell'accumulatore
214	ADC H	Somma del bit di carry e dei contenuti del registro H con i contenuti dell'accumulatore
215	ADC L	Somma del bit di carry e dei contenuti del registro L con i contenitori dell'accumulatore
216	ADC M	Somma del bit di carry e della locazione di memoria M con i contenuti dell'accumulatore
217	ADC A	Somma dei contenuti dell'accumulatore con i contenuti dell'accumulatore
220	SUB B	Sottrazione dei contenuti del registro B dai contenuti dell'accumulatore
221	SUB C	Sottrazione dei contenuti del registro C dai contenuti dell'accumulatore
222	SUB D	Sottrazione dei contenuti del registro D dai contenuti dell'accumulatore
223	SUB E	Sottrazione dei contenuti del registro E dai contenuti dell'accumulatore
224	SUB H	Sottrazione dei contenuti del registro H dai contenuti dell'accumulatore
225	SUB L	Sottrazione dei contenuti del registro L dai contenuti dell'accumulatore
226	SUB M	Sottrazione dei contenuti della locazione di memoria M dai contenuti dell'accumulatore
227	SUB A	Sottrazione dei contenuti dell'accumulatore dai contenuti dell'accumulatore (equivale all'azzerramento dell'accumulatore)
230	SBB B	Sottrazione del bit di carry e dei contenuti del registro B dai contenuti dell'accumulatore
231	SBB C	Sottrazione del bit di carry e dei contenuti del registro C dai contenuti dell'accumulatore
232	SBB D	Sottrazione del bit di carry e dei contenuti del registro D dai contenuti dell'accumulatore
233	SBB E	Sottrazione del bit di carry e dei contenuti del registro E dai contenuti dell'accumulatore
234	SBB H	Sottrazione del bit di carry e dei contenuti del registro H dai contenuti dell'accumulatore
235	SBB L	Sottrazione del bit di carry e dei contenuti del registro L dai contenuti dell'accumulatore
236	SBB M	Sottrazione del bit di carry e dei contenuti della locazione di memoria M dai contenuti dell'accumulatore
237	SBB A	Sottrazione del bit di carry e dei contenuti dell'accumulatore dai contenuti dell'accumulatore
004	INR B	Incremento di 1 dei contenuti del registro B
014	INR C	Incremento di 1 dei contenuti del registro C
024	INR D	Incremento di 1 dei contenuti del registro D
034	INR E	Incremento di 1 dei contenuti del registro E
044	INR H	Incremento di 1 dei contenuti del registro H
054	INR L	Incremento di 1 dei contenuti del registro L
064	INR M	Incremento di 1 dei contenuti della locazione di memoria M
074	INR A	Incremento di 1 dei contenuti dell'accumulatore
005	DCR B	Decremento di 1 dei contenuti del registro B

015	DCR C	Decremento di 1 dei contenuti del registro C
025	DCR D	Decremento di 1 dei contenuti del registro D
035	DCR E	Decremento di 1 dei contenuti del registro E
045	DCR H	Decremento di 1 dei contenuti del registro H
055	DCR L	Decremento di 1 dei contenuti del registro L
065	DCR M	Decremento di 1 dei contenuti della locazione di memoria M
075	DCR A	Decremento di 1 dei contenuti dell'accumulatore
003	INX B	Incremento di 1 dei contenuti della coppia reg. B e C
023	INX D	Incremento di 1 dei contenuti della coppia reg. D e E
043	INX H	Incremento di 1 dei contenuti della coppia reg. H e L
063	INX SP	Incremento di 1 dei contenuti del registro stack pointer
013	DCX B	Decremento di 1 dei contenuti della coppia reg. B e C
033	DCX D	Decremento di 1 dei contenuti della coppia reg. D e E
053	DCX H	Decremento di 1 dei contenuti della coppia reg. H e L
073	DCX SP	Decremento di 1 dei contenuti del registro stack pointer
011	DAD B	Somma dei contenuti della coppia di registri B e C ai contenuti della coppia di registri H ed L e memorizzazione dei risultati nella coppia H, L
031	DAD D	Somma dei contenuti della coppia di registri D e E ai contenuti della coppia di registri H ed L e memorizzazione dei risultati nella copia H, L
051	DAD H	Somma dei contenuti della coppia di registri H e L ai contenuti della coppia di registri H ed L e memorizzazione dei risultati nella coppia H, L
071	DAD SP	Somma dei contenuti dello stack pointer ai contenuti della coppia di registri H ed L e memorizzazione dei risultati nella coppia H, L
047	DAA	Aggiustamento decimale del numero a 8 bit contenuto nell'accumulatore, per formare due digit BCD a 4 bit (usare poi una istruzione addizionale perché aggiunge due numeri BCD)
306	ADI <B2>	Somma immediata del byte B2 con i contenuti dell'accumulatore
316	ACI <B2>	Somma immediata del bit di carry e del byte B2 con i contenuti dell'accumulatore
326	SUI <B2>	Sottrazione immediata del byte B2 dai contenuti dell'accumulatore
336	SBI <B2>	Sottrazione del byte di carry e immediata del byte B2 dai contenuti dell'accumulatore
270	CMP B	Confronto dei contenuti del registro B con i contenuti dell'accumulatore; l'accumulatore resta inalterato. I flag di condizione sono settati nel caso di una sottrazione tra i contenuti di B e quelli dell'accumulatore
271	CMP C	Confronto dei contenuti del registro C con i contenuti dell'accumulatore
272	CMP D	Confronto dei contenuti del registro D con i contenuti dell'accumulatore
273	CMP E	Confronto dei contenuti del registro E con i contenuti dell'accumulatore
274	CMP H	Confronto dei contenuti del registro H con i contenuti dell'accumulatore
275	CMP L	Confronto dei contenuti del registro L con i contenuti dell'accumulatore
276	CMP M	Confronto dei contenuti della locazione di memoria M con i contenuti dell'accumulatore
277	CMP A	Confronto dei contenuti dell'accumulatore con i contenuti dell'accumulatore
376	CPI <B2>	Confronto immediato del byte B2 con i contenuti dell'accumulatore

Gruppo Logico

Questo gruppo di istruzioni realizza operazioni logiche su dati presenti nei registri, in memoria e sui flag di condizione. *Con alcune eccezioni, tutte le istruzioni di questo gruppo modificano il flag di zero, segno, parità, carry e carry ausiliario, secondo le regole standard.* Le sole eccezioni sono le istruzioni RLC, RRC, RAL, RAR, CMC e STC (modificato solo il flag di carry) e l'istruzione CMA (nessun flag modificato).

240	ANA B	AND tra i contenuti del reg. B con i contenuti dell'accumulatore
241	ANA C	AND tra i contenuti del reg. C con i contenuti dell'accumulatore
242	ANA D	AND tra i contenuti del reg. D con i contenuti dell'accumulatore
243	ANA E	AND tra i contenuti del reg. E con i contenuti dell'accumulatore
244	ANA H	AND tra i contenuti del reg. H con i contenuti dell'accumulatore
245	ANA L	AND tra i contenuti del reg. L con i contenuti dell'accumulatore
246	ANA M	AND tra i contenuti della locazione di memoria M con i contenuti dell'accumulatore
247	ANA A	AND tra i contenuti dell'accumulatore con i contenuti dell'accumulatore
250	XRA B	OR-esclusivo dei contenuti del registro B con i contenuti dell'accumulatore
251	XRA C	OR-esclusivo dei contenuti del registro C con i contenuti dell'accumulatore
252	XRA D	OR-esclusivo dei contenuti del registro D con i contenuti dell'accumulatore
253	XRA E	OR-esclusivo dei contenuti del registro E con i contenuti dell'accumulatore
254	XRA H	OR-esclusivo dei contenuti del registro H con i contenuti dell'accumulatore
255	XRA L	OR-esclusivo dei contenuti del registro L con i contenuti dell'accumulatore
256	XRA M	OR-esclusivo dei contenuti della locazione di memoria M con i contenuti dell'accumulatore
257	XRA A	OR-esclusivo dei contenuti dell'accumulatore con i contenuti dell'accumulatore
260	ORA B	OR dei contenuti del registro B con i contenuti dell'accumulatore
261	ORA C	OR dei contenuti del registro C con i contenuti dell'accumulatore
262	ORA D	OR dei contenuti del registro D con i contenuti dell'accumulatore
263	ORA E	OR dei contenuti del registro E con i contenuti dell'accumulatore
264	ORA H	OR dei contenuti del registro H con i contenuti dell'accumulatore
265	ORA L	OR dei contenuti del registro L con i contenuti dell'accumulatore
266	ORA M	OR dei contenuti della locazione di memoria M con i contenuti dell'accumulatore
267	ORA A	OR dei contenuti dell'accumulatore con i contenuti dell'accumulatore

007	RLC	Rotazione dei contenuti dell'accumulatore a sinistra di una posizione, il bit meno significativo e il flag di carry sono settati entrambi al valore che esce dal bit più significativo
017	RRC	Rotazione dei contenuti dell'accumulatore a destra di una posizione, il bit più significativo e il flag di carry sono settati entrambi al valore che esce dal bit meno significativo
027	RAL	Rotazione dei contenuti dell'accumulatore a sinistra di una posizione attraverso il flag di carry. Il bit meno significativo è settato eguale al bit di carry e il flag di carry è settato al valore che esce dal bit più significativo
037	RAR	Rotazione dei contenuti dell'accumulatore a destra di una posizione attraverso il flag di carry. Il bit più significativo è settato eguale al bit di carry ed il flag di carry è settato al valore che esce dal bit meno significativo
057	CMA	Complemento dell'accumulatore
067	STC	Set del flag di carry, cioè bit di carry a stato logico 1
077	CMC	Complemento del bit di carry
346	ANI <B2>	AND immediato del byte B2 con i contenuti dell'accumulatore
356	XRI <B2>	OR-esclusivo immediato del byte B2 con i contenuti dell'accumulatore
366	ORI <B2>	OR immediato del byte B2 con i contenuti dell'accumulatore

Gruppo di Branch

Questo gruppo di istruzioni altera la normale sequenzialità di un programma. *Nessun flag è modificato da alcuna istruzione di questo gruppo.* I due tipi di istruzioni sono:
a) condizionate, b) non condizionate o incondizionate. I trasferimenti incondizionati semplicemente realizzano una specifica operazione (alterazione) sul contenuto a 16 bit del registro program counter. I trasferimenti condizionati esaminano lo stato di uno dei quattro flag di processo (zero, segno, parità, carry) al fine di determinare se lo specifico branch deve attuarsi. Le condizioni che si possono specificare sono:

flag di carry ad 1 logico	flag di carry a 0 logico
flag di zero ad 1 logico	flag di zero a 0 logico
flag di segno ad 1 logico	flag di segno a 0 logico
flag di parità ad 1 logico	flag di parità a 0 logico

302	JNZ <B2> <B3>	Salto alla locazione di memoria indirizzata dai byte B2 e B3, se il flag di zero è a 0 logico
312	JZ <B2> <B3>	Salto alla locazione di memoria indirizzata dai byte B2 e B3, se il flag di zero è a 1 logico
322	JNC <B2> <B3>	Salto alla locazione di memoria indirizzata dai byte B2 e B3, se il flag di carry è a 0 logico
332	JC <B2> <B3>	Salto alla locazione di memoria indirizzata dai byte B2 e B3, se il flag di carry è a 1 logico
342	JPO <B2> <B3>	Salto alla locazione di memoria indirizzata dai byte B2 e B3, se il flag di parità è a 0 logico

352	JPE <B2> <B3>	Salto alla locazione di memoria indirizzata dai byte B2 e B3, se il flag di parità è a 1 logico
362	JP <B2> <B3>	Salto alla locazione di memoria indirizzata dai byte B2 e B3, se il flag di segno è a 0 logico
372	JM <B2> <B3>	Salto alla locazione di memoria indirizzata dai byte B2 e B3, se il flag di segno è a 1 logico
304	CNZ <B2> <B3>	Richiamo della subroutine posta alla locazione di memoria indirizzata dai byte B2 e B3, se il flag di zero è a 0 logico
314	CZ <B2> <B3>	Richiamo della subroutine posta alla locazione di memoria indirizzata dai byte B2 e B3, se il flag di zero è a 1 logico
324	CNC <B2> <B3>	Richiamo della subroutine posta alla locazione di memoria indirizzata dai byte B2 e B3, se il flag di carry è a 0 logico
334	CC <B2> <B3>	Richiamo della subroutine posta alla locazione di memoria indirizzata dai byte B2 e B3, se il flag di carry è a 1 logico
344	CPO <B2> <B3>	Richiamo della subroutine posta alla locazione di memoria indirizzata dai byte B2 e B3, se il flag di parità è a 0 logico
354	CPE <B2> <B3>	Richiamo della subroutine posta alla locazione di memoria indirizzata dai byte B2 e B3, se il flag di parità è a 1 logico
364	CP <B2> <B3>	Richiamo della subroutine posta alla locazione di memoria indirizzata dai byte B2 e B3, se il flag di segno è a 0 logico
374	CM <B2> <B3>	Richiamo della subroutine posta alla locazione di memoria indirizzata dai byte B2 e B3, se il flag di segno è a 1 logico
300	RNZ	Ritorno da subroutine, se il flag di zero è a 0 logico
310	RZ	Ritorno da subroutine, se il flag di zero è a 1 logico
320	RNC	Ritorno da subroutine, se il flag di carry è a 0 logico
330	RC	Ritorno da subroutine, se il flag di carry è a 1 logico
340	RPO	Ritorno da subroutine, se il flag di parità è a 0 logico
350	RPE	Ritorno da subroutine, se il flag di parità è a 1 logico
360	RP	Ritorno da subroutine, se il flag di segno è a 0 logico
370	RM	Ritorno da subroutine, se il flag di segno è a 1 logico
303	JMP <B2> <B3>	Salto incondizionato alla locazione di memoria indirizzata da B2 e B3
311	RET	Ritorno da subroutine incondizionato
315	CALL <B2> <B3>	Richiamo incondizionato della subroutine posta alla locazione di memoria indirizzata dai byte B2 e B3
307	RST 0	Richiamo della subroutine a HI = 000_8 e LO = 000_8
317	RST 1	Richiamo della subroutine a HI = 000_8 e LO = 010_8
327	RST 2	Richiamo della subroutine a HI = 000_8 e LO = 020_8
337	RST 3	Richiamo della subroutine a HI = 000_8 e LO = 030_8

347	RST 4	Richiamo della subroutine a HI = 000_8 e LO = 040_8
357	RST 5	Richiamo della subroutine a HI = 000_8 e LO = 050_8
367	RST 6	Richiamo della subroutine a HI = 000_8 e LO = 060_8
377	RST 7	Richiamo della subroutine a HI = 000_8 e LO = 070_8
351	PCHL	Trasferimento dei contenuti di H e L nel program Counter, cioè salto indiretto alla locazione di memoria M indirizzata dalla coppia di registri H ed L

Gruppo di Stack, I/O, Controllo Macchina

Questo gruppo di istruzioni realizza operazioni di I/O, gestisce lo stack e modifica i flag di controllo. *Con una eccezione, nessun flag è alterato dalle istruzioni di questo gruppo.* La sola eccezione è la POP PSW, che modifica tutti i flag.

333	IN <B2>	Sostituisce i contenuti di A con un byte dati ad 8 bit proveniente da un dispositivo di ingresso selezionato dal codice dispositivo dati nel byte B2
323	OUT <B2>	Invia i contenuti dell'accumulatore, come dato ad 8 bit, verso un dispositivo di uscita selezionato dal codice dispositivo dati nel byte B2
373	EI	Abilitazione del sistema di interrupt *dopo l'esecuzione dell'istruzione successiva*
363	DI	Disabilitazione del sistema di interrupt *dopo l'esecuzione di questa istruzione*
166	HLT	Alt del microprocessore. I registri ed i flag non sono modificati
000	NOP	No operation (Nessuna operazione). I registri ed i flag non sono modificati
343	XTHL	Scambio del top dello stack con i contenuti della coppia di registri H ed L. Il contenuto di L è scambiato con quello della locazione di memoria SP, il cui indirizzo è dato dallo Stack Pointer Il contenuto di H è scambiato con quello della locazione di memoria SP + 1
301	POP B	Estrazione dallo stack e trasferimento nella coppia di registri B e C. Il contenuto della locazione di memoria SP è posto in C e quello della locazione di memoria SP + 1, in B. Lo stack pointer è incrementato di due.
321	POP D	Estrazione dallo stack e trasferimento nella coppia di registri D e E. Il contenuto della locazione di memoria SP è posto in E e quello della locazione di memoria SP + 1, in D. Lo stack pointer è incrementato di due
341	POP H	Estrazione dallo stack e trasferimento nella coppia di registri H e L. Il contenuto della locazione di memoria SP è posto in L e quello della locazione di memoria SP + 1, in H. Lo stack pointer è incrementato di due
361	POP PSW	Estrazione dallo stack e trasferimento del contenuto nell'accumulatore e ripristino dei flag di condizione. Il contenuto della locazione di memoria SP ripristina i flag di condizione e quello della locazione di memoria SP + 1 è posto nell'accumulatore. Lo stack pointer è incrementato di due. I byte HI e LO della coppia di registri PSW sono:

Byte Accumulatore	S Z 0 AC 0 P 1 C

Le lettere S, Z, AC, P e C indicano: segno, zero, carry ausiliario, parità e carry

305	PUSH B	Caricamento nello stack della coppia di registri B e C, rispettivamente nella locazione di memoria SP-1 ed SP-2. Lo stack pointer è decrementato di due
325	PUSH D	Caricamento nello stack della coppia di registri D e E, rispettivamente nella locazione di memoria SP-1 ed SP-2. Lo stack pointer è decrementato di due
345	PUSH H	Caricamento nello stack della coppia di registri H e L, rispettivamente nella locazione di memoria SP-1 ed SP-2. Lo stack pointer è decrementato di due
365	PUSH PSW	Caricamento nello stack dell'accumulatore e dei flag di condizione, rispettivamente nella locazione di memoria SP-1 ed SP-2. Lo stack pointer è decrementato a due

SOMMARIO DELLE ISTRUZIONI DELL'8080

Istruzioni a un solo Byte

INR r	0S4	INX B	003	POP B	301	RNZ	300	XCHG	353
DCR r	0S5	INX D	023	POP D	321	RZ	310	XTHL	343
		INX H	043	POP H	341	RNC	320	SPHL	371
MOV r_1r_2	1DS	INX SP	063	POP PSW	361	RC	330	PCHL	351
ADD r	20S	DCX B	013	PUSH B	305	RPO	340	HLT	166
ADC r	21S	DCX D	033	PUSH D	325	RPE	350	NOP	000
SUB r	22S	DCX H	053	PUSH H	345	RP	360	DI	363
SBB 4	23S	DCX SP	073	PUSH PSW	365	RM	370	EI	373
ANA r	24S					RET	311	DAA	047
XRA r	25S	DAD B	011	STAX B	002	RLC	007	CMA	057
ORA r	26S	DAD D	031	STAX D	022	RRC	017	STC	067
CMP r	27S	DAD H	051	LDAX B	012	RAL	027	CMC	077
		DAD SP	071	LDAX D	032	RAR	037	RST	3X7

S e D: B = 0, C = 1, D = 2, E = 3, H = 4, L = 5, M = 6, accumulatore = 7
X: 0 fino a 7

Istruzioni a due Byte

ADI <B2>	306	IN <B2>	333	MVI B <B2>	006
ACI <B2>	316	OUT <B2>	323	MVI C <B2>	016
SUI <B2>	326			MVI D <B2>	026
SBI <B2>	336			MVI E <B2>	036
ANI <B2>	346			MVI H <B2>	046
XRI <B2>	356			MVI L <B2>	056
ORI <B2>	366			MVI M <B2>	066
CPI <B2>	376			MVI A <B2>	076

Istruzioni a tre Byte

JNZ <B2> <B3>	302	CNZ <B2> <B3>	304	LXI B <B2> <B3>	001
JZ <B2> <B3>	312	CZ <B2> <B3>	314	LXI D <B2> <B3>	021
JNC <B2> <B3>	322	CNC <B2> <B3>	324	LXI H <B2> <B3>	041
JC <B2> <B3>	332	CC <B2> <B3>	334	LXI SP <B2> <B3>	061
JPO <B2> <B3>	342	CPO <B2> <B3>	344		
JPE <B2> <B3>	352	CPE <B2> <B3>	354	STA <B2> <B3>	062
JP <B2> <B3>	362	CP <B2> <B3>	364	LDA <B2> <B3>	072
JM <B2> <B3>	372	CM <B2> <B3>	374	SHLD <B2> <B3>	042
JMP <B2> <B3>	303	CALL <B2> <B3>	315	LHLD <B2> <B3>	052

LINGUAGGIO ASSEMBLER

Una volta che inizierete a programmare in linguaggio macchina binario, ottale o esadecimale, imparerete presto che la cosa può presentare delle difficoltà. Il problema sta nel fatto che molte istruzioni in linguaggio macchina fanno riferimento a specifiche locazioni in memoria. Se volete apportare dei cambiamenti al programma, potete aver bisogno di cambiare le locazioni di memoria allo scopo di sistemare delle nuove istruzioni inserite in punti specifici del programma originale. Un modo di aggirare il problema è quello di scrivere i vostri programmi in *linguaggio Assembler.*

Il termine «assembler» si riferisce al processo tramite il quale le istruzioni scritte in forma simbolica dal programmatore vengono cambiate in linguaggio macchina da un computer. I termini ad esso associati sono:

To assemble (Assemblare)	Tradurre da simbolico a binario, sostituendo codici operativi binari ai codici simbolici e sostituendo agli indirizzi simbolici gli indirizzi assoluti rilocabili.[4]
Assembler (Assemblatore)	Un programma che genera un programma in linguaggio macchina a partire da un programma scritto in linguaggio simbolico sostituendo codici operativi binari ai codici simbolici e indirizzi assoluti e rilocabili agli indirizzi simbolici.[4]
Assembly language (Linguaggio assembler)	Un linguaggio che ha una corrispondenza uno ad uno con un programma assembler. Il programma assemblatore porta un computer ad operare su di un programma in linguaggio simbolico per produrre un programma in linguaggio macchina.[4]
Assembly language programming, symbolic language programming (Programmazione in linguaggio assembler, programmazione in linguaggio simbolico)	La scrittura di un programma in un linguaggio che facilita la traslazione di programmi in codice binario tramite l'uso di istruzioni mnemoniche, quali ADD, MPY, SUB, DIV, STO, ecc.[4]
Assembly program (Programma assemblatore)	Un programma che abilita un computer ad assemblare il linguaggio mnemoico in linguaggio macchina. Chiamato anche «assembly routine».[4]

Symbolic address (Indirizzo simbolico)	Detto anche floating address. Si sceglie una label per identificare una particolare parola, funzione, o altre informazioni indipendenti dalla locazione dell'informazione nella routine.
Symbolic code (Codice simbolico)	Un codice tramite il quale vengono scritti i programmi in linguaggio sorgente, cioè si fa riferimento alle locazioni di memorizzazione e alle operazioni macchina con nomi simbolici e indirizzi simbolici che non dipendono dai loro nomi ed indirizzi legati alla struttura hardware.[4]
Symbolic coding (Codifica simbolica)	Nella programmazione dei computer digitali, qualunque sistema di codifica che usa indirizzi simbolici al posto di indirizzi reali.[4]
Symbolic programming (Programmazione simbolica)	Un programma che usa simboli al posto di numeri per le operazioni e le locazioni in un computer. Sebbene la scrittura del programma sia più facile e più veloce, bisogna usare un programma assemblatore per decodificare il simbolo in linguaggio macchina e per assegnare le locazioni delle istruzioni.[4]

E' importante capire la differenza fra linguaggi macchina e linguaggi assembler. Nella programmazione in linguaggio macchina:

- Ogni istruzione del programma è un numero binario specifico o un gruppo di numeri binari.
- Alle locazioni di memorizzazione di dati ed informazioni vengono assegnati indirizzi binari specifici in memoria.
- Le subroutine sono posizionate in specifici indirizzi binari di memoria.
- Ogni istruzione del programma è posizionata in un indirizzo di memoria binario specifico.
- Una volta scritto, il programma è pronto per partire.
- Come alternativa all'indirizzamento in memoria binario, si può usare l'indirizzamento in memoria ottale o esadecimale.
- I cambiamenti nel programma sono difficoltosi. Si possono richiedere cambiamenti negli indirizzi binari delle istruzioni, nei dati o nelle subroutine.

Nella programmazione in linguaggio assembler:

- Ogni istruzione del programma è un singolo codice mnemonico o un gruppo di codici mnemonici.
- Alle locazioni di memorizzazione dei dati e delle informazioni vengono assegnati anche nomi simbolici (o mnemonici).
- Le subroutine sono posizionate agli indirizzi dati da nomi mnemonici, chiamati *label* (etichetta).

- Ogni istruzione del programma è posizionata ad un indirizzo caratterizzato da una *label.*
- Una volta scritto, il programma non è pronto per essere avviato. Deve passare attraverso un assemblatore per convertire il codice simbolico (o mnemonico) in linguaggio macchina.
- Di solito, non si usa l'indirizzamento in memoria binario, ottale o esadecimale, anche se potrebbe essere conveniente.
- I cambiamenti nel programma vengono effettuati con relativa facilità. Dato che non è stato settato nessuno degli indirizzi di memoria per le istruzioni, i dati o le subroutine, si possono inserire passi nuovi in qualunque punto del programma in linguaggio assembler senza far sorgere difficoltà.

Nel prossimo paragrafo, esamineremo le differenze fra le istruzioni in linguaggio macchina e quelle in linguaggio assembler.

LINGUAGGIO MACCHINA E PROGRAMMI IN LINGUAGGIO ASSEMBLER

Scriveremo ora un semplice programma sia nel linguaggio macchina dell'8080 che in quello assembler. Questo è il programma tipico che useremo nei capitoli successivi.

- Sottrarre i contenuti dell'accumulatore dai contenuti dell'accumulatore, cioè azzerare l'accumulatore.
- Inviare un impulso di selezione dispositivo ed i contenuti dell'accumulatore ad un dispositivo chiamato «PRINTER» (stampante), che ha un codice dispositivo 000_8.
- Saltare incondizionatamente alla prima istruzione del programma, che è alla locazione di memoria chiamata «START», che è alla locazione di memoria HI = 000_8 e LO = 000_8.

La codifica ottale usata nel programma in linguaggio macchina è la seguente:

Indirizzo di memoria LO	*Istruzione ottale*	*Descrizione*
000_8	227	Sottrai i contenuti dell'accumulatore dai contenuti dell'accumulatore, cioè azzera l'accumulatore
001_8	323	Invia un impulso di selezione dispositivo ed i contenuti dell'accumulatore al dispositivo dato nel secondo byte dell'istruzione
002_8	000	Codice dispositivo
003_8	303	Salta incondizionatamente alla locazione di memoria data dai due byte seguenti
004_8	000	Byte dell'indirizzo di memoria LO
005_8	000	Byte dell'indirizzo di memoria HI

E il programma in linguaggio assembler è:

```
           *000 000
START,     SUB A
           OUT PRINTER
           JMP START
           START
PRINTER,   000
$
```

dove le *label* sono:

START = indirizzo di memoria a cui il programma inizia,
PRINTER = nome del dispositivo di uscita.

Le definizioni dei termini «*label*» e «*operando*» sono le seguenti:

Label	Uno o più caratteri che servono a definire un byte di dati, la posizione di un'istruzione o di una subroutine, o un dispositivo d'ingresso o di uscita.
Operand (Operando)	La quantità che viene coinvolta, manipolata, prodotta, o su cui si opera.

Per mettere in rilievo un punto già trattato, *nella programmazione in linguaggio assembler, vogliamo essere in grado di indirizzare le locazioni dei dati, i byte di dati, i dispositivi, le istruzioni e le subroutine con nomi simbolici invece che con codici ottali a 8 o a 16 bit.*

Notate che, nel programma in linguaggio assembler, non vi sono codici ottali. Tutte le istruzioni sono scritte sotto forma dei loro codici mnemonici e il dispositivo di uscita sotto forma di nome, PRINTER (stampante). La posizione d'inizio del programma si chiama START invece di HI = 000_8 e LO = 000_8. L'uso della programmazione in linguaggio assembler è notevolmente più facile della programmazione in linguaggio macchina, ma quest'ultima ha un vantaggio importante: è pronta per essere eseguita sul microcomputer, mentre il programma in linguaggio assembler non lo è. Con un programma in linguaggio assembler, bisogna fare uso di una tabella ed assegnare i valori degli indirizzi alle label e i valori dei codici operativi agli operandi. Si può «assemblare a mano» un programma assegnando i valori degli indirizzi e scegliendo gli equivalenti ottali per i codici mnemonici. I computer si adattano benissimo a questo compito con l'uso di programmi assembler e cross assembler.

INTRODUZIONE AGLI ESEMPI

Dato che la programmazione è una delle pratiche importanti che dovete conoscere a fondo se volete effettivamente usare i micro-

computer, vi forniamo vari esempi di programmazione che provano alcune delle caratteristiche del set di istruzioni dell'8080. Vengono messe in risalto alcune delle istruzioni più insolite, quali PUSH, POP e DAA, insieme a molti programmi di uscita che potete usare con un circuito latch/display a 8 bit, che imparerete a costruire nel Capitolo 7.

ESEMPIO N. 1

Scopo

Lo scopo di questo esempio è di mostrare in che modo si azzera la memoria del microcomputer.

Programma

Indirizzo di memoria LO	*Istruzione ottale*	*Codice mnemonico*	*Descrizione*
000	227	SUB A	Azzera l'accumulatore
001	041	LXI H	Carica i due byte di dati seguenti nei registri L e H, rispettivamente
002	011		Byte di dati L
003	000		Byte di dati H
004	167	MOV M, A	Trasferisci i contenuti dell'accumulatore nella locazione di memoria indirizzata dalla coppia di registri H ed L
005	043	INX H	Incrementa di uno la coppia di registri H e L
006	303	JMP	Salto incondizionato alla locazione di memoria data nei due byte seguenti:
007	004		Byte d'indirizzo di memoria LO
010	000		Byte d'indirizzo di memoria HI

Commenti

Questo programma vi permette di azzerare la memoria di lettura/scrittura, essenzialmente tutte le 65.536 locazioni di memoria se lo volete, partendo da HI = 000 e LO = 011. Non appena azzera la locazione HI = 377 e LO = 377, inizia ad azzerare il programma stesso. Arriva fino alla locazione di memoria HI = 000 e LO = 004. Infine, crea un loop indefinito fra le locazioni LO = 004 e LO = 010.

L'istruzione NOP è 000. Perciò, una locazione di memoria di lettura/scrittura azzerata, se trattata come un byte istruzioni, non eseguirà nessuna operazione utile. Azzerando in precedenza la memoria di lettura/scrittura, potete evitare l'esecuzione indesiderata del programma quando state provando un programma nuovo. [Un problema è costituito dal fatto che di solito non avete 65K di memoria nel microcomputer. Una locazione di memoria inesistente fornisce al chip del microprocessore 8080 un 377; quando viene trattata come un byte istruzioni, l'8080 chiama una subroutine alla locazione HI = 000 e LO = 070. Per minimizzare un problema di questo genere, potreste caricare il byte istruzioni 166 (HLT) a HI = 000 e LO = 070]. Vorremmo farvi notare, comunque

che non è necessario azzerare la memoria prima che venga usata dal microcomputer.

ESEMPIO N. 2

Scopo

Lo scopo di questo esempio è tentare di leggere i contenuti dei registri B e C, che sono settati ai valori 010_8 e 001_8, rispettivamente.

Programma

Indirizzo di memoria LO	*Istruzione ottale*	*Codice mnemonico*	*Descrizione*
000	041	LXI H	Carica i due byte di dati seguenti nei registri L e H, rispettivamente
001	200		Byte di dati L
002	000		Byte di dati H
003	001	LXI B	Carica i due byte di dati seguenti nei registri C e B, rispettivamente
004	001		Byte di dati C
005	010		Byte di dati B
006	166	HLT	Alt

Commenti

Dovreste essere in grado di eseguire con successo questo programma, ma non potete sapere se è accaduto qualcosa dal momento che tutti i dati restano all'interno del chip dell'8080. Studiate l'esempio seguente, il n. 3, per trovare uno dei modi possibili di visualizzare i contenuti dei registri interni.

ESEMPIO N. 3

Scopo

Lo scopo di questo esempio è quello di scrivere in memoria i contenuti dei registri B e C, che sono settati ai valori 010_8 e 001_8, rispettivamente.

Programma

Indirizzo di memoria LO	*Istruzione ottale*	*Codice mnemonico*	*Descrizione*
000	041	LXI H	Carica i due byte di dati seguenti nei registri L e H, rispettivamente
001	200		Byte di dati L
002	000		Byte di dati H
003	001	LXI B	Carica i due byte di dati seguenti nei registri C e B, rispettivamente
004	001		Byte di dati C
005	010		Byte di dati B
006	160	MOV M, B	Trasferisci i contenuti del registro B nella locazione di memoria indirizzata dalla coppia di registri H e L

007	043	INX H	Incrementa di uno la coppia di registri H e L
010	161	MOV M, C	Trasferisci i contenuti del registro C nella locazione di memoria indirizzata dalla coppia di registri H e L
011	166	HLT	Alt

Commenti

Con l'esempio n. 2, avete imparato che *è impossibile osservare direttamente i contenuti dei registri B, C, D, E, H, L.* Potete osservare i contenuti dell'accumulatore più o meno direttamente con l'aiuto di un'istruzione di OUT e di una coppia di latch (lo farete nel Capitolo 7). Comunque, dovete ricorrere a qualche trucco per determinare i contenuti dei sei registri universali. Ecco come fare:

- Memorizzare i contenuti dei registri, poi esaminare i contenuti della memoria mentre il microcomputer è nello stato di HOLD.
- Spingere i contenuti dei registri nello stack, poi esaminare i contenuti delle locazioni di memoria nello stack mentre il microprocessore è nello stato di HOLD.
- Trasferire i contenuti di ogni registro nell'accumulatore, poi fornire un'istruzione di OUT ed effettuare un latch sui contenuti dell'accumulatore. Potete farlo a turno con ogni registro, e potete seguire i contenuti del registro mentre il microcomputer sta ancora lavorando.

Sarebbe bello avere un gruppo di sette registri a LED, ognuno dei quali contenesse otto bit, per visualizzare continuamente i contenuti dei sei registri universali e dell'accumulatore. Con il microprocessore 8080, non è possibile farlo come uscite dirette dal chip.

ESEMPIO N. 4

Scopo

Lo scopo di questo esempio è quello di caricare nella regione dello stack in memoria i contenuti dei registri B e C, che sono settati ai valori 010_8 e 011_8. rispettivamente.

Programma

Indirizzo di memoria LO	*Istruzione ottale*	*Codice mnemonico*	*Descrizione*
000	061	LXI SP	Carica i due byte di dati seguenti nel registro stack pointer
001	202		Byte dello stack pointer LO
002	000		Byte dello stack pointer HI
003	001	LXI B	Carica i due byte di dati seguenti nei registri C e B, rispettivamente
004	001		Byte di dati C
005	010		Byte di dati B

006	305	PUSH B	Sostituisci i contenuti delle locazioni di memoria M (SP —1) e M (SP —2) con i contenuti dei registri B e C, rispettivamente
007	166	HLT	Alt

Commenti

Questo programma carica i contenuti del registro B nella locazione di memoria HI = 000 e LO = 201, e i contenuti del registro C nella locazione di memoria HI = 000 e LO = 200. Lo stack pointer è decrementato di uno prima che i contenuti di ogni registro vengano spinti sullo stack. Una volta che il programma si è arrestato, potete indirizzare le posizioni dello stack e dimostrare che i contenuti dei registri sono stati memorizzati in quelle locazioni.

ESEMPIO N. 5

Scopo

Scopo di questo esempio è estrarre i dati memorizzati in precedenza dalla regione dello stack in memoria, incrementare e decrementare i contenuti dei dati estratti, e poi riporli nello stack. Si usa la coppia di registri B.

Programma

Indirizzo di memoria LO	*Istruzione ottale*	*Codice mnemonico*	*Descrizione*
000	061	LXI SP	Carica i due byte di dati seguenti nel registro stack pointer
001	200		Byte dello stack pointer LO
002	000		Byte dello stack pointer HI
003	227	SUB A	Azzera l'accumulatore
004	301	POP B	Sostituisci i contenuti del registro B e C con i contenuti delle locazioni dello stack M (SP + 1) e M (SP), rispettivamente
005	004	INR B	Incrementa di uno i contenuti del registro B
006	015	DCR C	Decrementa di uno i contenuti del registro C
007	305	PUSH B	Sostituisci i contenuti delle locazioni di memoria M (SP —1) e M (SP —2) con i contenuti dei registri B e C, rispettivamente (Nota; ricordatevi che lo stack pointer è stato incrementato di due come risultato dell'istruzione POP B all'indirizzo 004_8)
010	166	HLT	Alt

Dati

Indirizzo di memoria LO	*Dati in memoria*
200	222
201	333

Commenti

Quando eseguirete questo programma, osserverete il byte ottale 221 nelle locazioni di memoria HI = 000 e LO = 200, e il byte ottale 334 nelle locazioni di memoria HI = 000 e LO = 201. I contenuti iniziali del registro B sono memorizzati a LO = 201; i contenuti iniziali del registro C sono memorizzati a LO = 200. Il registro B viene incrementato, e il registro C decrementato. Lo stack pointer verrà prima incrementato di due, e poi decrementato di due. Alla fine del programma, lo stack pointer sarà a HI = 000 e LO = 200.

ESEMPIO N. 6

Scopo

Scopo di questo esempio è visualizzare lo stato dei cinque flag, dopo una semplice operazione aritmetica, con l'aiuto dell'istruzione PUSH PSW.

Programma

Indirizzo di memoria LO	*Istruzione ottale*	*Codice mnemonico*	*Descrizione*
000	061	LXI SP	Carica i due byte di dati seguenti nel registro stack pointer
001	202		Byte dello stack pointer LO
002	000		Byte dello stack pointer HI
003	227	SUB A	Sottrai i contenuti dell'accumulatore dai contenuti dell'accumulatore, cioè azzera l'accumulatore
004	365	PUSH PSW	Sostituisci i contenuti delle locazioni di memoria M (SP —1) e M (SP —2) con i contenuti dell'accumulatore e con lo stato dei cinque flag di condizione, rispettivamente
005	166	HLT	Alt

Commenti

Questo è un programma utile che vi permette di provare in che modo le istruzioni aritmetiche e logiche influenzano i cinque flag: segno, zero, carry ausiliario, parità e carry. Dopo che avete eseguito un'operazione a LO = 003, ponete i contenuti dei flag e dell'accumulatore sullo stack. Dopo che il microprocessore si è arrestato, potete arrivare direttamente alla regione dello stack e determinare lo stato dei flag. La parola di stato del programma a 16 bit appare in questo modo sullo stack:

Byte Accumulatore	S	Z	0	AC	0	P	1	C

Il byte dell'accumulatore è il byte del registro HI ed è memorizzato a LO = 201. Il byte dei flag è memorizzato a LO = 200. Le lettere S, Z, AC, P, C nell'illustrazione precedente si riferiscono ai flag di segno, di zero, di carry ausiliario, di parità e di carry.

Quando eseguite questo programma ed esaminate la locazione di memoria LO = 200, osserverete il byte a 8 bit 01010110, che significa quanto segue:

Z = 0 Il risultato è positivo
S = 1 Il risultato è zero
AC = 1 (Senza significato)
P = 1 Il risultato ha parità pari
C = 0 Il risultato non ha carry

Se modifificate i byte istruzioni a LO = 004 fino a LO = 006 come vi mostreremo ora e poi eseguite il programma, osserverete il byte a 8 bit 00000010 a LO = 200.

Indirizzo di memoria LO	*Istruzione ottale*	*Codice mnemonico*	*Descrizione*
004	074	INR A	Incrementa i contenuti dell'accumulatore di uno
005	365	PUSH PSW	Inserisci i contenuti dell'accumulatore e dei flag nello stack
006	166	HLT	Alt

Il byte dell'accumulatore a LO = 201 sarà 001. Questi risultati significano quanto segue:

S = 0 Il risultato è positivo
Z = 0 Il risultato non è zero
AC = 0 Il risultato non ha carry dal bit D3 al bit D4 dell'accumulatore
P = 0 Il risultato ha parità dispari
C = 0 Il risultato non ha carry

Questo tipo di programma vi mostra lo stato dei flag individuali prima che tentiate di usare le istruzioni di salto condizionato. Tutte le operazioni logiche e la maggior parte di quelle aritmetiche alterano i flag. Una volta che sapete come si comporterà un flag, potete usare le istruzioni logico/aritmetiche e le istruzioni di salto condizionato appropriate del vostro programma principale.

ESEMPIO N. 7

Scopo

Scopo di questo esempio è esplorare le istruzioni di salto condizionato e determinare quando avviene un salto. (Vedi Fig. 3-14).

Programma

Indirizzo di memoria LO	*Istruzione ottale*	*Codice mnemonico*	*Descrizione*
000	061	LXI SP	Carica i due byte di dati seguenti nel registro stack pointer
001	200		Byte dello stack pointer LO
002	000		Byte dello stack pointer HI
003	361	POP PSW	Estrai i contenuti dell'accumulatore e i flag dallo stack
004	*	*	Codice operativo per una qualunque delle otto istruzioni condizionate (JNZ, JZ, JNC, JC, JPO, JPE, JP, o JM)
005	004		Byte dell'indirizzo di memoria LO
006	000		Byte dell'indirizzo di memoria HI
007	166	HLT	Alt
200	*	*	Byte di dati del flag che verrà estratto dallo stack
201	000		Byte di dati dell'accumulatore che verrà estratto dallo stack

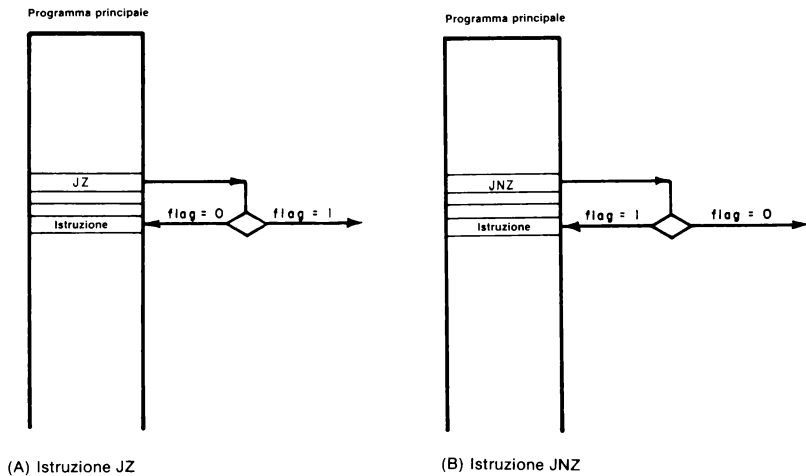



Fig. 3-14. Istruzioni di salto condizionato JZ e JNZ

Prima di eseguire questo programma, caricate la memoria con una parola di stato, il byte dei flag che si trova a HI = 000 e LO = 200, e il byte di dati dell'accumulatore a HI = 000 e LO = 201. Caricate anche un'istruzione di salto condizionato desiderata all'indirizzo di memoria HI = 000 e LO = 004. Avendo fatto tutto questo, l'esecuzione del programma vi permette di provare in che modo i bit dei vari flag influenzano l'esecuzione di una qualunque delle otto istruzioni di salto condizionato.

Per dimostrare il programma, potete confrontare le istruzioni di salto condizionato JNZ e JZ, che sono rappresentate schematicamente nella Fig. 3-14.

Con un byte dei flag di **002** a LO = 200 e il byte dell'istruzione JNZ, **302,** a LO = 004, dovreste osservare che il programma eseguito compie dei loop fra LO = 004 e LO = 006. Il flag di segno, S, è a livello logico 0, ed indica all'8080 che si verifica la condizione di «non zero»; avviene quindi un salto a HI = 000 e LO = 004. Se cambiate il byte dei flag da **002** in **102,** non avverrà nessun salto e il programma si arresterà.

Con un byte di flag di **002** e il byte dell'istruzione JZ, dovreste osservare che il programma si arresta. Quando cambiate i flag in **102,** il programma effettua un loop fra LO = 004 e LO = 006. Questo comportamento si può prevedere dalla natura dell'istruzione JZ, in cui avviene un salto solo se il flag di zero, Z, è a livello logico 1.

Possiamo riassumere il comportamento del programma per diversi byte di flag estratti e diverse istruzioni di salto condizionato, come mostra l'elenco seguente.

Byte di flag a LO = 200	*Byte di flag interessato*	*Stato logico del byte di flag*	*Istruzione di salto condizionato*	*Comportamento del programma*
002	Z	0	JNZ	loop
102	Z	1	JNZ	alt
002	Z	0	JZ	alt
102	Z	1	JZ	loop
002	C	0	JNC	loop
003	C	1	JNC	alt
002	C	0	JC	alt
003	C	1	JC	loop
002	P	0	JPO	loop
006	P	1	JPO	alt
002	P	0	JPE	alt
006	P	1	JPE	loop
002	S	0	JP	loop
202	S	1	JP	alt
002	S	0	JM	alt
202	S	1	JM	loop

In ognuno dei casi dell'elenco precedente, abbiamo provato uno dei bit di flag per l'istruzione di salto condizionato indicata.

ESEMPIO N. 8

Scopo

Scopo di questo esempio è mostrare l'esecuzione di un programma che contiene un «nido» di subroutine.

Programma N. 1

Indirizzo di memoria LO	*Istruzione ottale*	*Codice mnemonico*	*Descrizione*
000	061	LXI SP	Carica lo stack pointer con l'indirizzo dato dai due byte seguenti
001	003	HLT	Byte d'indirizzo LO dello stack pointer
002	003		Byte d'indirizzo HI dello stack pointer
003	166		Alt

Il diagramma di flusso di questo programma è mostrato nella Fig. 3-15.

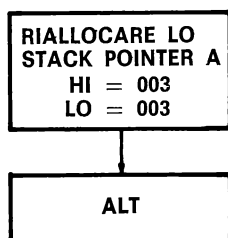



Fig. 3-15. Diagramma di flusso del Programma N. 1

Programma N. 2

Dopo aver eseguito il Programma n. 1, caricate il programma principale, che inizia a HI = 000.

Indirizzo di memoria LO	*Istruzione ottale*	*Codice mnemonico*	*Descrizione*
000	317	RST 1	Chiama la subroutine a HI = 000 e LO = 010
001	166	HLT	Alt
010	327	RST 2	Chiama la subroutine a HI = 000 e LO = 020
011	311	RET	Ritorna dalla subroutine
020	337	RST 3	Chiama la subroutine a HI = 000 e LO = 030
021	311	RET	Ritorna dalla subroutine
030	000	NOP	Nessuna operazione
031	311	RET	Ritorna dalla subroutine

Il diagramma di flusso di questo programma è mostrato nella Fig. 3-16.

Commenti

Dato che questo esempio dimostra come il microcomputer tratta le subroutine nidificate, *vi diamo in dettaglio l'esecuzione del programma.*

Byte d'indirizzo HI	LO	Bus di dati	Codice mnemonico	Descrizione
000	000	317	RST 1	Memorizza il contatore di programma nello stack, poi salta alla subroutine a LO = 010
003	002	000		Byte d'indirizzo HI del contatore di programma
003	001	001		Byte d'indirizzo LO del contatore di programma
000	010	327	RST 2	Memorizza il contatore di programma nello stack, poi salta alla subroutine a LO = 020
003	000	000		Byte d'indirizzo HI del contatore di programma
002	377	011		Byte d'indirizzo LO del contatore di programma
000	020	337	RST 3	Memorizza il contatore di programma nello stack, poi salta alla subroutine a LO = 030
002	376	000		Byte d'indirizzo HI del contatore di programma
002	375	021		Byte d'indirizzo LO del contatore di programma
000	030	000	NOP	Nessuna operazione
000	031	311	RET	Estrai il programma dallo stack, cioè ritorna dalla subroutine 5

Soffermiamoci su questo punto dell'esecuzione del programma ed esaminiamo che cosa è successo fin qui. Abbiamo chiamato tre subroutine. Tre set di byte del contatore di programma sono stati memorizzati sullo stack e lo stack pointer è stato fatto scorrere verso il basso. Abbiamo eseguito ora la nostra istruzione RET e

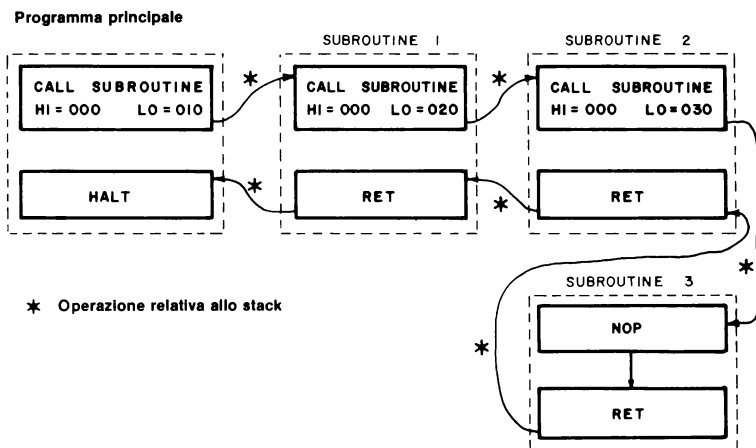



Fig. 3-16. Diagramma di flusso di Programma N. 2

siamo pronti per osservare i byte del contatore di programma che vengono estratti dallo stack.

002	375	021		Byte d'indirizzo LO del contatore di programma
002	376	000		Byte d'indirizzo HI del contatore di programma
000	021	311	RET	Estrai il contatore di programma dallo stack, cioè ritorna dalla subroutine 2
002	377	011		Byte d'indirizzo LO del contatore di programma
003	000	000		Byte d'indirizzo HI del contatore di programma
000	011	311	RET	Estrai il contatore di programma dallo stack, cioè ritorna dalla subroutine 1
003	001	001		Byte d'indirizzo LO del contatore di programma
003	002	000		Byte d'indirizzo HI del contatore di programma
000	001	000	HLT	Alt

Il programma precedente dovrebbe convincervi che siete in grado di nidificare un numero qualsiasi di subroutine. L'unico vincolo consiste nel fornire una sufficiente memoria lettura/scrittura per lo stack. Il microcomputer gestirà tutte le informazioni associate con le istruzioni di chiamata di subroutine e di ritorno. Potete ora riferirvi al diagramma di flusso del programma N. 2.

ESEMPIO N. 9

Scopo

Scopo di questo esempio è dimostrare in che modo potete determinare le conseguenze di diverse istruzioni dell'accumulatore.

Alcune Istruzioni dell'Accumulatore dell'8080

Molte delle istruzioni dell'8080 coinvolgono il registro accumulatore, conosciuto anche come registro A. In questo esempio, vi forniamo un semplice programma che vi permetterà di provare varie istruzioni dell'accumulatore, tra cui le seguenti:

Istruzione ottale	*Codice mnemonico*	*Descrizione*
007	RLC	Fai ruotare i contenuti dell'accumulatore di una posizione verso sinistra
017	RRC	Fai ruotare i contenuti dell'accumulatore di una posizione verso destra
027	RAL	Fai ruotare i contenuti dell'accumulatore di una posizione verso sinistra attraverso il flag di riporto
037	RAR	Fai ruotare i contenuti dell'accumulatore di una posizione verso destra attraverso il flag di riporto

047	DAA	Aggiustamento decimale dell'accumulatore
057	CMA	Complementa i contenuti dell'accumulatore
072 <B2> <B3>	LDA	Carica l'accumulatore in modo diretto con i contenuti dell'indirizo di memoria dato dai byte d'indirizzo <B2> e <B3>
074	INR A	Incrementa di uno i contenuti dell'accumulatore
075	DCR A	Decrementa di uno i contenuti dell'accumulatore
076 <B2>	MVI A	Carica l'accumulatore con il byte a 8 bit immediatamente successivo
170	MOV A, B	Trasferisci i contenuti del registro B nell'accumulatore
171	MOV A, C	Trasferisci i contenuti del registro C nell'accumulatore
172	MOV A, D	Trasferisci i contenuti del registro D nell'accumulatore
173	MOV A, E	Trasferisci i contenuti del registro E nell'accumulatore
174	MOV A, H	Trasferisci i contenuti del registro H nell'accumulatore
175	MOV A, L	Trasferisci i contenuti del registro L nell'accumulatore
176	MOV A, M	Trasferisci i contenuti della locazione di memoria, il cui indirizzo è dato dalla coppia di registri H ed L, nell'accumulatore
177	MOV A, A	Trasferisci i contenuti dell'accumulatore nell'accumulatore
200	ADD B	Somma i contenuti del registro B ai contenuti dell'accumulatore
206	ADD M	Somma i contenuti della locazione di memoria, il cui indirizzo è dato dalla coppia di registri H ed L, ai contenuti dell'accumulatore
207	ADD A	Somma i contenuti dell'accumulatore ai contenuti dell'accumulatore
210	ADC B	Somma il bit di carry e i contenuti del registro B ai contenuti dell'accumulatore
217	ADC A	Somma il bit di carry e i contenuti dell'accumulatore ai contenuti dell'accumulatore
220	SUB B	Sottrai i contenuti del registro B dai contenuti dell'accumulatore
227	SUB A	Azzera l'accumulatore
230	SBB B	Sottrai il bit di carry e i contenuti del registro B dai contenuti dell'accumulatore
240	ANA B	Esegui un'operazione di AND sui contenuti del registro B con i contenuti dell'accumulatore
247	ANA A	Esegui un'operazione di AND sui contenuti dell'accumulatore con i contenuti dell'accumulatore
250	XRA B	Esegui un'operazione di OR Esclusivo sui contenuti del registro B con i contenuti dell'accumulatore
257	XRA A	Azzera l'accumulatore
260	ORA B	Esegui un'operazione di OR sui contenuti del registro B con i contenuti dell'accumulatore
267	ORA A	Esegui un'operazione di OR sui contenuti dell'accumulatore con i contenuti dell'accumulatore
270	CMP B	Confronta i contenuti del registro B con i contenuti dell'accumulatore

276		CMP M	Confronta i contenuti della locazione di memoria, il cui indirizzo è dato dalla coppia di registri H ed L, con i contenuti dell'accumulatore
277		CMP A	Confronta i contenuti dell'accumulatore con i contenuti dell'accumulatore
306	<B2>	ADI	Somma il byte a 8 bit immediatamente successivo ai contenuti dell'accumulatore
316	<B2>	ACI	Somma il byte a 8 bit immediatamente successivo e il bit di carry ai contenuti dell'accumulatore
326	<B2>	SUI	Sottrai il byte a 8 bit immediatamente successivo ai contenuti dell'accumulatore
336	<B2>	SBI	Sottrai il byte a 8 bit immediatamente successivo e il bit di carry dai contenuti dell'accumulatore
346	<B2>	ANI	Esegui un'operazione di AND sul byte a 8 bit immediatamente successivo con i contenuti dell'accumulatore
356	<B2>	XRI	Esegui un'operazione di OR Esclusivo sui byte immediatamente successivi con i contenuti dell'accumulatore
361		POP PSW	Estrai il top dello stack e memorizzane i contenuti nell'accumulatore e nei flip-flop dei flag
366	<B2>	ORI	Esegui un'operazione di OR sul byte a 8 bit immediatamente successivo con i contenuti dell'accumulatore
376	<B2>	CPI	Confronta il byte a 8 bit immediatamente successivo con i contenuti dell'accumulatore

Programma

Indirizzo di memoria LO	*Istruzione ottale*	*Codice mnemonico*	*Descrizione*
000	061	LXI SP	Carica i due byte di dati successivi nel registro stack pointer
001	000		Byte dello stack pointer LO
002	001		Byte dello stack pointer HI
003	027	SUB A	Azzera l'accumulatore
004	016	MVI C	Trasferisci il byte di dati seguente nel registro C
005	002		Byte di timing per il registro C
006	315	CALL	Chiama in modo incondizionato la subroutine posizionata all'indirizzo di memoria dato dai due byte seguenti
007	100		Byte d'indirizzo di memoria LO
010	000		Byte d'indirizzo di memoria HI
011	007	RLC	Fai ruotare i contenuti dell'accumulatore di una posizione verso sinistra
012	074	INR A	Incrementa di uno i contenuti dell'accumulatore
013	000	NOP	Nessuna operazione
014	000	NOP	Nessuna operazione
015	000	NOP	Nessuna operazione
016	323	OUT	Genera un impulso di selezione dispositivo che permetta ad un latch a 8 bit di venire applicato ai contenuti dell'accumulatore
017	000	000	Codice dispositivo per il latch a 8 bit

020	303	JMP	Salto incondizionato alla posizione data dai due byte d'indirizzo seguenti
021	004		Byte d'indirizzo LO
022	000		Byte d'indirizzo HI

Subroutine (Genera un Ritardo di Tempo)

100	021	LXI D	Trasferisci i due byte seguenti nei registri E e D, rispettivamente
101	301		Byte di timing per il registro E
102	150		Byte di timing per il registro D
103	035	DCR E	Decrementa i contenuti del registro E di uno
104	302	JNZ	Se il registro E è 000_8, ignora questa istruzione; altrimenti, salta all'indirizzo di memoria dato nei due byte seguenti
105	103		Byte d'indirizzo LO
106	000		Byte d'indirizzo HI
107	025	DCR D	Decrementa di uno i contenuti del registro D
110	302	JNZ	Se il registro D è 000_8, ignora questa istruzione; altrimenti, salta alla locazione di memoria data nei due byte seguenti
111	103		Byte d'indirizzo LO
112	000		Byte d'indirizzo HI
113	015	DCR C	Decrementa di uno i contenuti del registro C
014	302	JNZ	Se il registro C è 000_8, ignora questa istruzione, altrimenti salta alla locazione di memoria data dai due byte seguenti
115	100		Byte d'indirizzo LO
116	000		Byte d'indirizzo HI
117	311	RET	Ritorna in modo incondizionato da questa subroutine

Commenti

Questo programma richiede un latch di uscita a 8 bit e un display per permettervi di osservare i risultati delle operazioni aritmetiche e logiche che eseguite sui contenuti dell'accumulatore. Fate riferimento ai circuiti descritti nel Capitolo 7.

La subroutine che inizia a HI = 000 e LO = 100 genera ritardi di tempo che vanno da 0,200 secondi a 51,2 secondi attraverso variazioni nel byte di timing per il registro C, all'indirizzo di memoria LO di 005.

Se eseguite il programma come si trova, osserverete che il display di uscita si riempie in fretta di livelli logici 1, da destra verso sinistra. Agli indirizzi di memoria da 011 a 015, avete cinque byte di programma con cui potete eseguire diversi tipi di operazioni dell'accumulatore. Quindi, con il segmento di programma:

011	076	MVI A	Carica il byte seguente nell'accumulatore
012	360	360	Byte di dati che corrisponde alla parola binaria 11110000_8
013	346	ANI	Esegui un'operazione di AND sui dati seguenti con i contenuti dell'accumulatore
014	252	252	Byte di dati che corrisponde alla parola binaria 10101010_2
015	000	NOP	Nessuna operazione

dovreste osservare che l'operazione AND fra il byte 11110000 e il byte 10101010 produce il risultato logico 10100000, un'operazione che procede bit per bit. Cambiando l'istruzione logica a LO = 013, potete dimostrare il comportamento delle istruzioni di OR e di OR Esclusivo sugli stessi dati iniziali.

Se eseguite il segmento di programma seguente contenuto all'interno del programma principale:

011	074	INR A	Incrementa i contenuti dell'accumulatore di uno
012	067	STC	Setta il flag di carry a livello logico 1
013	077	CMC	Complementa il flag di carry
014	047	DAA	Aggiusta l'accumulatore secondo il sistema decimale
015	000	NOP	Nessuna operazione

dovreste osservare un conteggio di uscita decimale da 0 a 99 sul display di uscita. L'istruzione **047** è l'istruzione di aggiustamento decimale dell'accumulatore, che converte il risultato dell'addizione di due numeri bcd in una coppia di numeri bcd impaccati. *Non* è un'istruzione di conversione da binario in bcd.

ESEMPIO N. 10

Scopo

Scopo di questo esempio è quello di dimostrare l'ingresso e la conversione diretta BCD in routine binaria, che è il n. 80-147 nella «Intel Microcomputer User's Library». Questo programma è stato sviluppato da M. H. Gansler.

Programma

Indirizzo di memoria LO	*Istruzione ottale*	*Codice mnemonico*	*Descrizione*
000	076	MVI A	Trasferisci immediatamente il byte nell'accumulatore
001	*		Byte di dati a due cifre bcd che deve essere convertito in un numero binario a 8 bit
002	117	MOV C, A	Trasferisci i contenuti dell'accumulatore nel registro C

003	346	ANI	Esegui un'operazione di AND del byte immediato con i contenuti dell'accumulatore
004	017	017	Byte di maschera per mascherare la cifra bcd più significativa
005	137	MOV E, A	Trasferisci i contenuti dell'accumulatore nel registro E
006	171	MOV A, C	Trasferisci i contenuti del registro C nell'accumulatore
007	346	ANI	Esegui un'operazione di AND del byte immediato con i contenuti dell'accumulatore
010	360	360	Byte di maschera per mascherare la cifra bcd meno significativa
011	017	RRC	Fai ruotare i contenuti dell'accumulatore di un bit verso sinistra e nel flag di carry
012	017	RRC	Fai ruotare i contenuti dell'accumulatore di un bit verso destra e nel flag di carry
013	117	MOV C, A	Trasferisci i contenuti dell'accumulatore nel registro C
014	017	RRC	Fai ruotare i contenuti dell'accumulatore di un bit verso destra e nel flag di carry
015	017	RRC	Come sopra
016	201	ADD C	Somma i contenuti del registro C ai contenuti dell'accumulatore
017	007	RLC	Fai ruotare i contenuti dell'accumulatore di un bit verso sinistra e nel flag di carry
020	203	ADD E	Somma i contenuti del registro E ai contenuti dell'accumulatore
021	323	OUT	Metti in uscita i contenuti dell'accumulatore alla porta di uscita data nel byte istruzioni seguente
022	000	000	Codice dispositivo per la porta di uscita zero
023	166	HLT	Alt

Commenti

Il programma inizia con il numero a due cifre nell'accumulatore. Il risultato è memorizzato nell'accumulatore.

Per convertire il programma in una subroutine, sostituite l'istruzione RET con l'istruzione HLT a LO = 023. Il programma può essere rilocato in qualunque punto della memoria.

Se tentate di eseguire questo programma, potete confrontare i vostri risultati con quelli che seguono:

Numero decimale	*Numero binario osservato*
1	00000001
10	00001010
20	00010100

50	00110010
75	01001011
80	01010000
90	01011010
99	01100011

ESEMPIO N. 11

Scopo

Scopo di questo esempio è mostrare una subroutine di addizione a 16 cifre bcd, in cui vengono sommati fra loro due numeri bcd per produrre un risultato che è minore o uguale a 9.999.999.999.999.999. Questo programma è elencato e descritto nei minimi particolari nel *«Software Manual»* e viene presentato qui per gentile concessione della NEC Microcomputer Inc. Il programma viene iniziato alla locazione di memoria HI = 003 e LO = 024.

Programma

Indirizzo di memoria	*LO*	*Byte istruzioni*	*Codice mnemonico*	*Descrizione*
	024	021	LXI D	Carica due byte immediati nei registri E e D, rispettivamente
	025	347		I registri D ed E contengono l'indirizzo a 16 bit delle cifre meno significative
	026	003		nell'augendo
	027	041	LXI H	Carica due byte immediati nei registri L ed H, rispettivamente
	030	357		I registri H e L contengono l'indirizzo
	031	003		a 16 bit delle cifre meno significative nell'addendo
ADD16:	032	365	PUSH PSW	Carica la parola di stato nello stack (Nota: assicuratevi di aver caricato lo stack pointer prima di eseguire questo programma)
	033	305	PUSH B	Carica i contenuti della coppia di registri B, C nello stack
	034	016	MVI C	Trasferisci il byte immediato nel registro C
	035	010		Il numero binario uguale a una volta e mezza il numero di cifre bcd. Quindi, per 16 cifre bcd, il codice ottale sarebbe 010
	036	257	XRA A	Azzera l'accumulatore e il flag di carry
LOOP2:	037	032	LDAX D	Carica l'accumulatore dalla locazione di memoria indirizzata dalla coppia di registri D, E
	040	216	ADC M	Somma i contenuti della locazione di memoria indirizzata dalla coppia di registri H, L ai contenuti dell'accumulatore
	041	047	DAA	Aggiusta i contenuti dell'accumulatore secondo il sistema decimale

	042	022	STAX D	Memorizza i contenuti dell'accumulatore nella locazione di memoria indirizzata dalla coppia di registri D, E
	043	015	DCR C	Decrementa di uno i contenuti del registro C
	044	312	JZ	Salta alla locazione di memoria DONE2 se i contenuti del registro C sono zero
	045	054		Byte d'indirizzo LO di DONE 2
	046	003		Byte d'indirizzo HI di DONE 2
	047	053	DCX H	Decrementa di uno i contenuti della coppia di registri H, L
	050	033	DCX D	Decrementa di uno i contenuti della coppia di registri D, E
	051	303	JMP	Salta alla locazione di memoria LOOP2
	052	037		Byte d'indirizzo LO di LOOP2
	053	003		Byte d'indirizzo HI di LOOP2
DONE2:	054	301	POP B	Estrai i contenuti della coppia di registri B, C dallo stack
	055	361	POP PSW	Estrai la parola di stato dallo stack
	056	172	MOV A, D	Trasferisci i contenuti del registro D nell'accumulatore
	057	323	OUT	Metti in uscita i contenuti dell'accumulatore
	060	001	001	Codice dispositivo della porta 1
	061	173	MOV A, E	Trasferisci i contenuti del registro E nell'accumulatore
	062	323	OUT	Metti in uscita i contenuti dell'accumulatore
	063	000	000	Codice dispositivo della porta zero
	064	166	HLT	Alt

Commenti

Questo programma inizia con un augendo bcd 16 cifre negli indirizzi di memoria da 340 a 347, con la cifra bcd meno significativa nella posizione 347 e quella più significativa nella posizione 340. L'addendo bcd a 16 cifre è inizialmente negli indirizzi di memoria LO da 350 a 357, con la cifra bcd meno significativa nella posizione 357 e quella più significativa nella posizione 350. I termini addendo e augendo vengono così definiti:[2]

Augend (Augendo) — In un'addizione aritmetica, il numero che viene aumentato aggiungendovi un altro numero (chiamato addendo).

Addend (Addendo) — Una quantità che, se aggiunta ad un'altra quantità (chiamata augendo) produce un risultato chiamato somma.

L'esecuzione del programma inizia a HI = 003 e LO = 024. La somma sostituisce l'augendo.

Consideriamo un augendo di 1.000.000.000.000.099 e un addendo di 8.000.000.000.000.001. La mappa di memoria per questi due nu-

meri bcd a 16 cifre è la seguente (il tutto a HI = 003):

Indirizzo di memoria LO	*Numero BCD*	*Codice ottale*	*Codice binario*
340	1.0	020	00010000
341	0.0	000	00000000
342	0.0	000	00000000
343	0.0	000	00000000
344	0.0	000	00000000
345	0.0	000	00000000
346	0.0	000	00000000
347	9.9	231	10011001
350	8.0	200	10000000
351	0.0	000	00000000
352	0.0	000	00000000
353	0.0	000	00000000
354	0.0	000	00000000
355	0.0	000	00000000
356	0.0	000	00000000
357	0.1	001	00000001

Quando questi due numeri vengono sommati, la somma (9.000.000.000.000.100) sostituisce l'augendo nelle locazioni di memoria HI = 003 e LO = 340 fino a HI = 003 e LO = 347.

Dovreste osservare la seguente sequenza di numeri bcd nelle locazioni di memoria successiva partendo da LO = 340:

90
00
00
00
00
00
01
00

che corrispondono al numero bcd a 16 cifre 9.000.000.000.000.100.

Potete, volendo, sommare i seguenti numeri bcd e confrontare i vostri risultati con le somme previste.

Augendo	*Addendo*	*Somma*
3.000.000.000.000.100	1.000.000.000.000.001	4.000.000.000.000.101
0.000.000.000.123.456	0.000.000.000.240.833	0.000.000.000.364.289
0.000.000.000.927.928	0.000.000.000.844.992	0.000.000.001.772.920
9.999.999.999.999.999	0.000.000.000.000.001	0.000.000.000.000.000

ESEMPIO N. 12

Scopo

Scopo di questo esempio è spiegare la subroutine di conversione da binario a BCD, n. 8-67 della Intel Microcomputer User's Library. Il programma è stato sviluppato da Niels S. Gundestrup del Geophysical Isotope Laboratory in Danimarca.

Programma

Indirizzo di memoria	*LO*	*Byte istruzioni*	*Codice mnemonico*	*Descrizione*
	222	021	LXI D	Trasferisci i due byte immediati nella coppia di registri D. Questo è il numero binario a 16 bit che sarà convertito in un numero a 5 cifre bcd
	223	*		Otto bit meno significativi del numero binario a 16 bit
	224	*		Otto bit più significativi del numero binario a 16 bit
	225	041	LXI H	Trasferisci i due byte immediati nella coppia di registri H. Questo è l'indirizzo di memoria della cifra più significativa (MSD) del numero bcd a 5 cifre. Le quattro cifre rimanenti sono memorizzate nelle locazioni di memoria successive, una cifra per posizione
	226	340		Byte del registro L
	227	003		Byte del registro H
BNBCD:	230	365	PUSH PSW	Carica i contenuti della parola di stato nello stack
	231	305	PUSH B	Carica i contenuti della coppia di registri B nello stack
	232	325	PUSH D	Carica i contenuti della coppia di registri D nello stack
	233	345	PUSH H	Carica i contenuti della coppia di registri H nello stack
	234	353	XCHG	Scambia i contenuti della coppia di registri H con quelli della coppia di registri D
	235	001	LXI B	Trasferisci i due byte immediati nella coppia di registri B (10.000)
	236	360		Byte del registro C
	237	330		Byte del registro B
	240	315	CALL	Chiama la subroutine DECNO, che esegue la conversione da binario a bcd (MSD)
	241	276		Byte d'indirizzo LO
	242	003		Byte d'indirizzo HI
	243	001	LXI B	Trasferisci i due byte immediati nella coppia di registri B (1.000)
	244	030		Byte del registro C
	245	374		Byte del registro B
	246	315	CALL	Chiama la subroutine DECNO

```
          247   276                Byte d'indirizzo LO
          250   003                Byte d'indirizzo HI
          251   001   LXI B        Trasferisci i due byte immediati nella
                                   coppia di registri B (100)
          252   234                Byte del registro C
          253   377                Byte del registro B
          254   315   CALL         Chiama la subroutine DECNO
          255   276                Byte d'indirizzo LO
          256   003                Byte d'indirizzo HI
          257   001   LXI B        Trasferisci i due byte immediati nella
                                   coppia di registri B (10)
          260   366                Byte del registro C
          261   377                Byte del registro B
          262   315   CALL         Chiama la subroutine DECNO
          263   276                Byte d'indirizzo LO
          264   003                Byte d'indirizzo HI
          265   175   MOV A, L     Trasferisci i contenuti del registro L
                                   nell'accumulatore
          266   306   ADI          Somma il byte immediato ai contenuti
                                   dell'accumulatore
          267   000   000          (Nota: 260 se si vuole il codice ASCII)
          270   022   STAX D       Memorizza i contenuti dell'accumula-
                                   tore nella locazione di memoria indi-
                                   rizzata dalla coppia di registri D
          271   341   POP H        Estrai la coppia di registri H dallo stack
          272   321   POP D        Estrai la coppia di registri D dallo stack
          273   301   POP B        Estrai la coppia di registri B dallo stack
          274   361   POP PSW      Estrai la parola di stato dallo stack
          275   311   RET          Ritorna dalla subroutine
DECNO:    276   076   MVI A        Trasferisci il byte seguente nell'accu-
                                   mulatore
          277   000   000          (Nota: 260 se si vuole il codice ASCII)
          300   325   PUSH D       Carica il registro D sullo stack
          301   135   MOV E, L     Trasferisci i contenuti del registro L
                                   nel registro E
          302   124   MOV D, H     Trasferisci i contenuti del registro H
                                   nel registro D
          303   074   INR A        Incrementa di uno i contenuti dell'ac-
                                   cumulatore
          304   011   DAD B        Somma i contenuti della coppia di re-
                                   gistri B ai contenuti della coppia di
                                   registri H e memorizza nella coppia di
                                   registri H
          305   332   JC           Salta se il flag di carry è a livello logi-
                                   co 1
          306   301                Byte d'indirizzo LO
          307   003                Byte d'indirizzo HI
          310   075   DCR A        Decrementa di uno i contenuti dell'ac-
                                   cumulatore
          311   153   MOV L, E     Trasferisci i contenuti del registro E
                                   nel registro L
          312   142   MOV H, D     Trasferisci i contenuti del registro D
                                   nel registro H
          313   321   POP D        Estrai la coppia di registri D dallo stack
```

314	022	STAX D	Memorizza i contenuti dell'accumulatore nella locazione di memoria indirizzata dalla coppia di registri D
315	023	INX D	Incrementa di uno i contenuti della coppia di registri D
316	311	RET	Ritorna dalla subroutine DECNO

Commenti

Questo programma inizia con un numero binario a 16 bit nella coppia di registri D, E. Il numero viene convertito in un numero a 5 cifre binarie, che è memorizzato a partire da HI = 003 e LO = 340. La cifra bcd più significativa viene memorizzata a questa posizione, e le quattro cifre rimanenti nelle posizioni successive. La cifra bcd meno significativa è memorizzata a LO = 344. Il programma BNBCD inizia a HI = 003 e LO = 230; comunque, il numero binario a 16 bit deve trovarsi nella coppia di registri D, e la posizione della cifra più significativa nella coppia di registri H. Abbiamo usato istruzioni LXI per settare questa informazione nei registri prima che BNBCD sia eseguito.

L'uscita può essere costituita o da numeri decimali o da codici ASCII a 8 bit, con il bit più signifificativo (il bit di parità) a livello logico 1. E' stato corretto un piccolo errore nel programma originale per permettere a LSD di essere memorizzato in codice ASCII.

TEST

Questo test verifica quanto avete capito delle tecniche e dei concetti della programmazione che sono stati descritti in questo capitolo. Per favore scrivete le vostre risposte su un foglio a parte.

3.1 Spiegate la differenza fra le seguenti coppie di concetti:

Istruzione - operazione
Istruzione - programma
Istruzione in linguaggio assembler - istruzione mnemonica
Istruzione mnemonica - istruzione in linguaggio macchina
Codice macchina - codice mnemonico
Registro - coppia di registri
Contatore di programma - stack pointer
Bit - byte
Byte - parola
Byte d'indirizzo HI - byte d'indirizzo LO
Programma - subroutine
Routine - programma
Registro istruzioni - decodificatore di istruzioni
Salto - chiamata
Flag di zero - flag di segno
Flag di carry - flag di carry ausiliario

PUSH - POP
Salto condizionato - salto incondizionato
Label - operando
Flag di parità - flag di segno
Accumulatore - ALU
Byte di dati - byte d'indirizzo
Codice ottale - codice esadecimale
Incremento - decremento
OR - OR Esclusivo
Sottrazione - confronto
Stack - stack pointer
Istruzione IN - istruzione OUT
Coppia di registri B - coppia di registri H
Istruzione MOV - istruzione MVI
Istruzione ADD - istruzione ADC
Carry - carry negativo

3.2 Riassumere i cinque gruppi di istruzioni fondamentali e fornire esempi di istruzioni all'interno di ogni gruppo.

3.3 Scrivere un semplice programma per il microcomputer sia in codice macchina che in linguaggio assembler. Con il linguaggio assembler usare operandi e label.

3.4 Descrivere i cinque flag di condizione e gli stati logici che caratterizzano le cóndizioni specifiche.

3.5 Spiegare la differenza fra codice ottale e codice esadecimale, e dare parecchi esempi di byte di dati a 8 bit scritti in entrambi i tipi di codice.

3.6 Fare un esempio di come si comporta l'istruzione PUSH.

3.7 Fare un esempio di come si comporta l'istruzione POP.

3.8 Fare un elenco di almeno nove diversi modi di caricare i dati nell'accumulatore.

3.9 Descrivere i quattro diversi tipi di istruzioni di rotazione nel set di istruzioni del microprocessore 8080.

3.10 Quali istruzioni dell'8080 si userebbero per:
Azzerare l'accumulatore
Settare l'accumulatore a 11111111_2
Mettere in uscita i dati dall'accumulatore su di un dispositivo di uscita
Inserire i dati nell'accumulatore da un dispositivo d'ingresso
Trasferire i dati dal registro E nell'accumulatore
Trasferire i dati dall'accumulatore nella locazione di memoria HI $= 000_8$ e LO $= 200_8$
Trasferire i dati nell'accumulatore dalla locazione di memoria H $= 000_8$ e L $= 200_8$
Trasferire il byte di dati 10101110_2 nell'accumulatore
Controllare lo stato logico di ogni bit nell'accumulatore

Moltiplicare i contenuti dell'accumulatore per 4
Dividere i contenuti dell'accumulatore per 8
Memorizzare i contenuti dell'accumulatore in una locazione di memoria memorizzata nella coppia di registri B e C
Caricare l'accumulatore da una locazione di memoria memorizzata nella coppia di registri D ed E
Trasferire i contenuti dell'accumulatore ad una locazione di memoria
Settare il bit di riporto a livello logico 0.

Usare, in tutti questi casi, il minor numero di istruzioni possibile.

La vostra prova sarà accettabile se sarete in grado di rispondere correttamente a tutte le domande suddette, a libro chiuso e in quattro ore di tempo.

CHE COSA AVETE REALIZZATO IN QUESTO CAPITOLO?

All'inizio di questo capitolo, si era stabilito che, alla fine sareste stati in grado di:

- Spiegare qual'è la differenza fra istruzione, operazione, programma, istruzione in codice macchina, istruzione in linguaggio assembler e istruzione mnemonica.
  Avete imparato a distinguere i termini suddetti nel testo di questo capitolo.

- Definire i termini: assemblaggio, bit, byte, flag, simbolo mnemonico, codice dispositivo, byte d'indirizzo HI, byte d'indirizzo LO, incremento, decremento, label, salto, chiamata, ritorno, operando, flag di carry, flag di parità, flag di zero, flag di segno, registro, coppia di registri, subroutine, istruzione a due byte, istruzione a tre byte, operazione incondizionata, operazione condizionata, istruzione di salto, stack, stack pointer, contatore di programma, accumulatore, ALU, byte di dati e registro istruzioni.
  Le definizioni di questi termini sono state fornite all'inizio e nel corso di tutto il capitolo.

- Classificare le istruzioni dell'8080 in cinque gruppi.
  I cinque gruppi sono: trasferimento dati, aritmetiche, logiche, di salto e un gruppo finale composto da istruzioni di stack, I/O e istruzioni di controllo macchina.

- Spiegare come si può scrivere un'istruzione a 8 bit sia in codice ottale che in codice esadecimale.
  Di questo abbiamo parlato in un paragrafo di questo capitolo.

- Fare un elenco dei codici mnemonici, seguendo le raccomandazioni della Intel Corporation, per almeno dieci diverse istruzioni dell'8080.

Vi sono 78 istruzioni diverse, per cui questo obiettivo sarebbe piuttosto difficile da raggiungere.

- Spiegare la differenza fra linguaggio macchina e linguaggio assembler.
  Lo abbiamo fatto verso la fine di questo capitolo.

- Identificare il byte d'indirizzo HI e il byte d'indirizzo LO in una parola d'indirizzo di memoria a 16 bit.
  Non dovreste avere difficoltà in proposito.

- Spiegare la differenza fra un bit, un byte, una parola e un indirizzo.
  Vi abbiamo fornito le definizioni, riferendoci al microprocessore 8080.

- Fare un elenco di almeno dieci diversi registri che si possono trovare nel microprocessore 8080.
  Ve ne forniamo un elenco in questo capitolo. I registri più importanti sono l'accumulatore, B, C, D, E, H, L, lo stack pointer, e i registri contatori di programma, che sono nove in tutto.

- Spiegare in che modo il microprocessore decodifica:
  Le classi di istruzioni
  I registri
  Le coppie di registri
  Le operazioni immediate
  Le operazioni di salto
  I flag di condizione
  Le operazioni di incremento
  Le operazioni di decremento
  A questi argomenti, è stata dedicata la maggior parte del capitolo.

CAPITOLO 4

Come si Genera un Impulso di Selezione Dispositivo

In questo capitolo, imparerete come generare impulsi di selezione dispositivo da un microcomputer basato sull'8080. Questi impulsi verranno usati in successivi capitoli di questo Bugbook per effettuare un latch sui dati in uscita ed anche per permettere ai dati di essere inseriti nell'accumulatore. Il microprocessore 8080 ha prestazioni notevoli dato che può generare fino a 256 diversi impulsi di selezione dispositivi di uscita e 256 diversi impulsi di selezione dispositivi di entrata. Questo dovrebbe essere più che sufficiente per qualunque applicazione ragionevole del microprocessore 8080.

OBIETTIVI

Alla fine di questo capitolo, sarete in grado di:

- Identificare l'istruzione OUT e l'istruzione IN in un programma del microprocessore 8080.
- Tracciare lo schema di un circuito che può generare fino a 256 diversi impulsi di selezione dispositivi.
- Spiegare come il microprocessore 8080 genera gli impulsi di selezione dispositivo.
- Scrivere semplici programmi che usano le istruzioni IN o OUT.
- Fare lo schema a blocchi per un decodificatore 74154, da quattro a sedici linee.

DEFINIZIONI

Device select pulse (Impulso di selezione dispositivo) — Un impulso di clock negativo o positivo generato dal software di un computer, usato per fornire impulsi di strobe ad uno o più dispositivi di I/O, compresi singoli circuiti integrati.

I/O *(I/O)* — Abbreviazione di input-output (ingresso uscita).

I/O device *(Dispositivo di I/O)* — Qualunque dispositivo digitale, compreso un circuito integrato, che trasmette dati o impulsi di strobe ad un computer, o viceversa che riceve dati o impulsi di strobe da un computer.

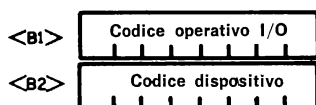
Machine cycle *(Ciclo macchina)* — Una suddivisione di un ciclo istruzioni, tempo richiesto per eseguire un'istruzione completa. Un ciclo macchina è il periodo più breve richiesto per eseguire un gruppo di azioni collegate durante l'esecuzione di un ciclo di istruzioni.

ISTRUZIONI DI I/O DEL MICROPROCESSORE 8080

Vi sono solo due istruzioni di I/O nel microcomputer 8080:

333 <B2> IN — Genera un impulso di selezione dispositivo, per fare sì che un byte di dati a 8 bit venga letto dal dispositivo d'ingresso e *sostituisca i contenuti dell'accumulatore.*

323 <B2> OUT — Genera un impulso di selezione dispositivo per fare sì che un byte di dati a 8 bit presente nell'accumulatore *venga mandato ad un dispositivo di uscita. I contenuti dell'accumulatore restano invariati.*

Queste due istruzioni hanno la forma seguente:

dove il *codice dispositivo* è un byte a 8 bit che specifica uno fra 256 dispositivi diversi. Entrambe queste istruzioni si assomigliano molto. L'unica differenza fra di loro riguarda quello che succede nell'accumulatore. Con l'istruzione d'ingresso, i contenuti dell'accumulatore cambiano, mentre con l'istruzione di uscita, questo non succede.

Il termine «impulso di selezione dispositivo» può essere così definito:

Impulso di selezione dispositivo. Un impulso di clock, positivo o negativo, generato dal software di un computer, usato per fornire impulsi di strobe ad uno o più dispositivi di I/O, compresi singoli circuiti integrati.

L'importanza degli impulsi di selezione dispositivo è determinata dall'uso che se ne fa per fornire impulsi di strobe per i dati in ingresso nell'accumulatore durante un'istruzione d'ingresso o per fornire impulsi di strobe per il latch dei dati in uscita dall'accu-

mulatore durante un'istruzione di uscita. Comunque, possono essere usati anche *per fornire impulsi di strobe per le operazioni di dispositivi di I/O in condizioni in cui non avvengano trasferimenti di dati da o verso l'accumulatore.* Così, come abbiamo precisato nel Capitolo 1, gli impulsi di selezione dispositivo sono singoli impulsi di clock che si possono usare in molti modi diversi, per esempio, per simulare il comportamento di un astabile 555, di un generatore d'impulsi, o di un monostabile 74121 o 555.

I codici mnemonici per le due istruzioni suddette sono IN <B2> e OUT <B2>, ed ognuno richiede dieci cicli di clock, o 5 μs, per essere eseguita. Nessuna delle due istruzioni coinvolge qualcuno dei cinque flag di condizione del microprocessore 8080. Il simbolo <B2> indica che deve esserci un secondo byte di istruzione, in questo caso un indirizzo dispositivo, nel programma, che segue immediatamente il codice istruzioni IN o OUT.

LA DECODIFICAZIONE DEGLI IMPULSI DI SELEZIONE DISPOSITIVO

I 256 diversi impulsi di selezione dispositivo vengono attuati con l'aiuto degli otto bit meno significativi nella parola d'indirizzo di memoria a 16 bit, cioè il byte d'indirizzo di memoria LO, o gli otto bit più significativi nella parola d'indirizzo di memoria, cioè il byte d'indirizzo HI. *Entrambi i byte a 8 bit vanno bene per generare il codice dispositivo nel microprocessore 8080;* noi abbiamo usato il byte d'indirizzo LO nei nostri esempi.

I decodificatori, come il decodificatore 74154 da quattro a sedici linee, la cui configurazione dei pin e lo schema a blocchi sono nella Fig. 4-1, decodificano il codice dispositivo a 8 bit in 256 diversi impulsi di selezione dispositivo. Un solo decodificatore può fornire sedici impulsi diversi. Quattro bit del codice dispositivo a 8 bit vengono applicati agli ingressi da A a D dal pin 23 al pin 20.

(A) Configurazione dei pin

(B) Schema a blocchi

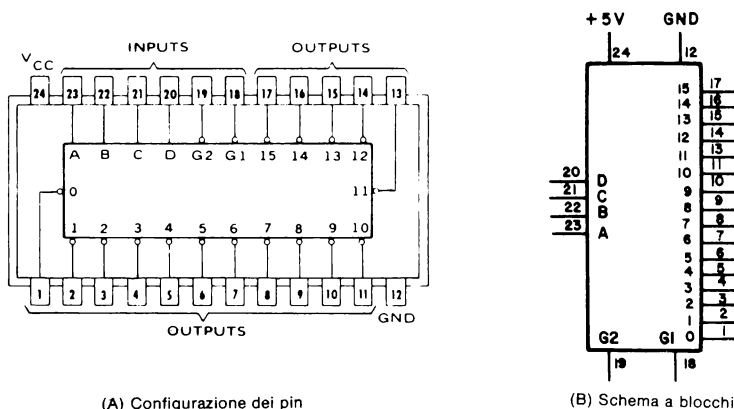



Fig. 4-1. Il decodificatore 74154 da quattro a sedici linee.

Le sedici uscite si ottengono ai pin da 1 a 11, da 13 a 17. I due ingressi di strobe, G1 e G2, devono essere entrambi a livello logico 0; le quindici uscite rimanenti sono tutte a livello logico 1.

Un Decodificatore 74154, Sedici Impulsi di Selezione Dispositivo

Si possono usare molte strategie diverse per decodificare il codice dispositivo a 8 bit e generare impulsi individuali di selezione dispositivo. Il circuito di decodifica più semplice è mostrato nella Fig. 4-2: vengono generati sedici diversi impulsi di selezione dispositivo con l'aiuto dei quattro bit meno significativi nella parola d'indirizzo di memoria LO e l'impulso di sincronizzazione $\overline{IN}$ o $\overline{OUT}$ del microcomputer 8080. L'impulso $\overline{IN}$, o $\overline{OUT}$, fornisce lo strobe al decodificatore, a G1; G2 è legato al livello logico 0. L'impulso di selezione dispositivo generato con questo circuito è un impulso di clock negativo, che per alcuni usi deve essere invertito con l'aiuto di un invertitore esadecimale 7404.

La maggior parte dei semplici circuiti di interfaccia dei microcomputer non richiederanno più di sedici impulsi di selezione dispositivo.

Codici dispositivo accettabili per lo schema della Fig. 4-2 comprendono quelli indicati di seguito, dove indica che, nella posizione del bit indicata, va bene sia un livello logico 0 che un livello logico 1:

XXXX0000	XXXX1000
XXXX0001	XXXX1001
XXXX0010	XXXX1010
XXXX0011	XXXX1011
XXXX0100	XXXX1100
XXXX0101	XXXX1101
XXXX0110	XXXX1110
XXXX0111	XXXX1111

Per i dispositivi di ingresso e di uscita, bisogna usare decodificatori separati. Così, una coppia di decodificatori 74154 vi per-

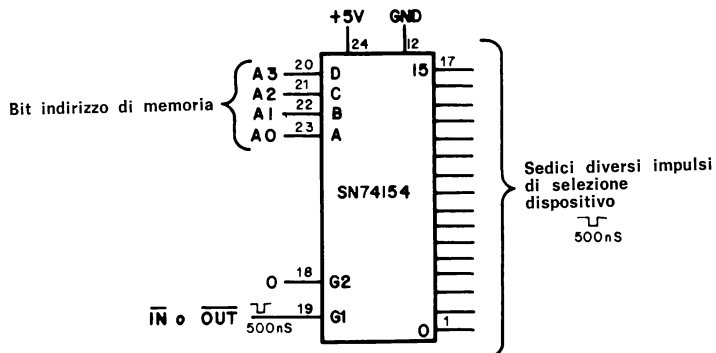



Fig. 4-2 Un semplice circuito di decodifica che può generare sedici diversi impulsi di selezione dispositivo. La decodificazione del bus degli indirizzi non è assoluta.

mette di selezionare fino a sedici diversi dispositivi d'ingresso e sedici diversi dispositivi di uscita.

Diciassette Decodificatori 74154, 256 Impulsi di Selezione Dispositivo

Nella Fig. 4-3, vedete un circuito che richiede un solo collegamento $\overline{IN}$ o $\overline{OUT}$ e che può generare fino a 256 diversi impulsi di selezione dispositivo. I quattro bit più significativi nella parola dell'indirizzo di memoria LO a 4 bit selezionano il decodificatore sulla destra, mentre i quattro bit meno significativi selezionano il canale di uscita specifico sul decodificatore selezionato. Tutti gli impulsi di selezione dispositivo sono impulsi di clock negativi.

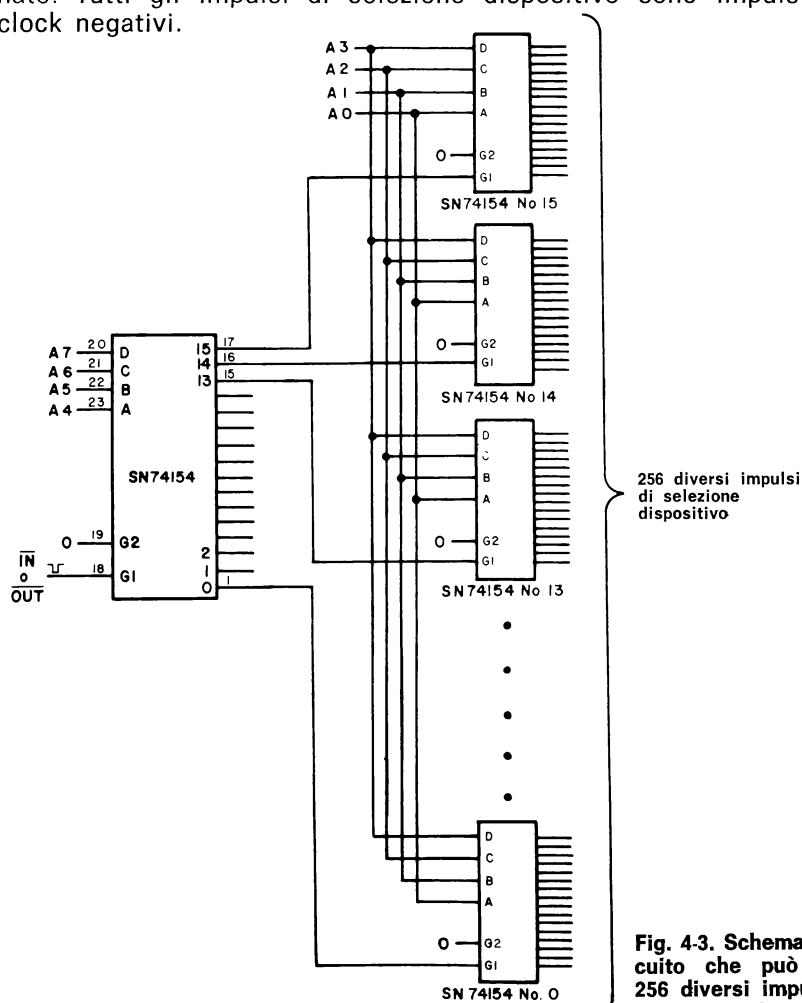



Fig. 4-3. Schema di un circuito che può generare 256 diversi impulsi di selezione dispositivo.

Altri Circuiti di Decodifica

Vi sono molti altri metodi che si possono usare per generare gli impulsi di selezione dispositivo OUT o IN associati con i dispositivi di I/O dei microcomputer. Mentre l'hardware può cambiare e può essere diverso da applicazione ad applicazione, il software è sempre lo stesso. Si specifica sempre un'istruzione OUT **(323)** o IN **(333)** e un codice dispositivo.

Il circuito di decodifica mostrato nella Fig. 4-4 dimostra come si possano usare due decodificatori 74154 e una porta NOR a due ingressi appropriati per generare anche un unico impulso di selezione dispositivo, uno su 256, per ogni codice dispositivo. Mentre questo circuito richiede meno decodificatori dello schema mostrato in Fig. 4-3, è necessario una porta NOR per ogni impulso di selezione dispositivo generato. Potreste usare anche por-

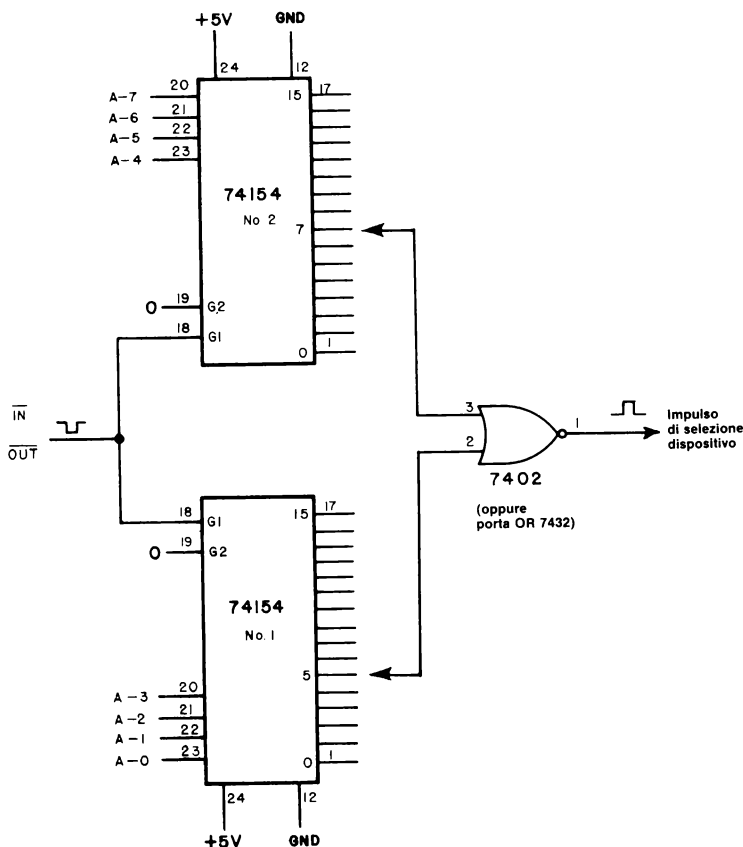



Fig. 4-4. Un altro circuito che può generare 256 diversi impulsi di selezione dispositivo. Potete vedere solo una delle 256 porte NOR richieste.

te OR, se aveste bisogno di impulsi di selezione dispositivo negativi anziché positivi. La Fig. 4-5 mostra come si possono generare gli impulsi di selezione dispositivi di uscita 000, 001, 002, 003 e 004; lo schema decodifica solo istruzioni di uscita. Dovete duplicare l'hardware della Fig. 4-5 per decodificare l'indirizzo dei dispositivi d'ingresso.

Gli schemi di decodifica precedenti si usano per generare impulsi di selezione dispositivo d'ingresso e di uscita allo scopo di trasferire dati o generare segnali di controllo quando alcuni dispositivi sono posti molto vicini uno all'altro, di solito sulla stessa piastra di circuito stampato. Le unità periferiche o lontane, di solito hanno un semplice decodificatore per il dispositivo specifico. Per esempio, se volete generare un unico indirizzo per un dispositivo, come 371, e un impulso di selezione dispositivo o d'ingresso o di uscita, dovreste usare il circuito dato nella Fig. 4-6. Potreste usare una coppia di comparatori 7485 per produrre un unico impulso di selezione dispositivo, come l'impulso di uscita $\overline{\text{DS 306}}$, solo quando tutti e due i set di otto ingressi corrispondono perfettamente. (Fig. 4-7). Questo è un circuito molto flessibile che viene usato in sistemi in cui bisogna cambiare il codice dispositivo.

Se avete bisogno di parecchi impulsi di selezione dispositivo, le combinazioni degli schemi e decodificatori precedenti sono effettivamente operative. Il circuito della Fig. 4-8 si può usare per generare gli indirizzi dei dispositivi 070, 071, 072 e 073. La porta

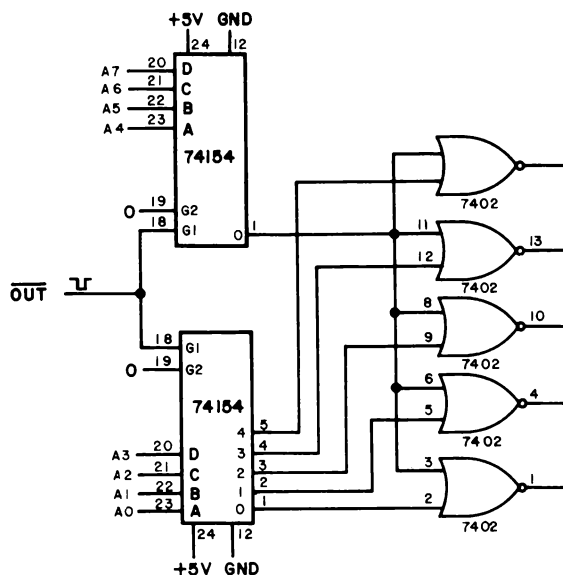



Fig. 4-5. Schema di decodifica per generare impulsi di selezione di uscita 000, 001, 002, 003 e 004.

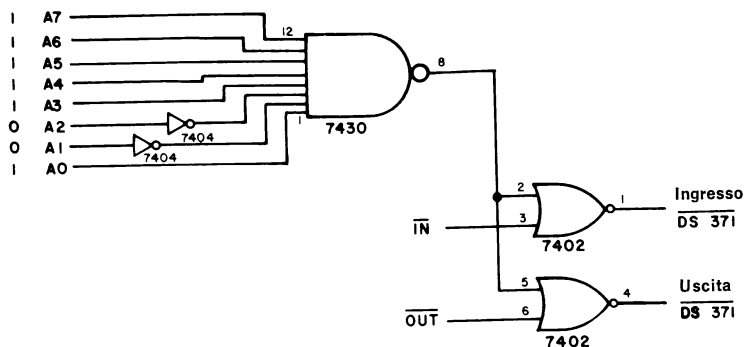



Fig. 4-6. Circuito di decodificazione in assoluto basata sull'uso di gate NAND a 8 ingressi 7430.

NAND 7430 a otto ingressi decodifica i bit d'indirizzo da A2 ad A7. Quando a questo gate appare la combinazione giusta di indirizzi d'ingresso, il decodificatore 7442 viene abilitato all'ingresso D (pin 12). L'ingresso C del chip 7442 si usa per un altro impulso di abilitazione, in questo caso o $\overline{\text{IN}}$ o $\overline{\text{OUT}}$.

Si sarebbe potuta usare una coppia di comparatori al posto del chip 7430 e degli invertitori 7404 associati.

Mentre l'hardware è cambiato nei circuiti di decodifica precedenti, il software di base necessario per generare gli indirizzi e gli impulsi $\overline{\text{IN}}$ o $\overline{\text{OUT}}$ rimane lo stesso.

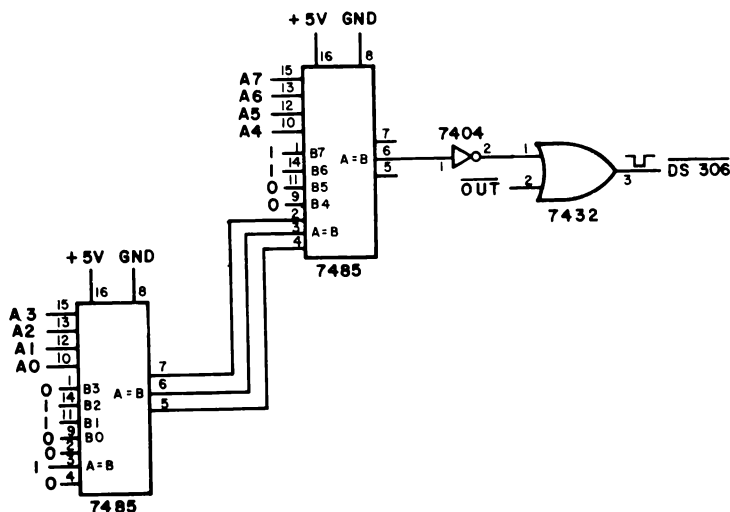



Fig. 4-7. Circuito di decodifica in assoluto basato sull'uso di una coppia di comparatori a 4 bit 7485.

UN ESEMPIO DI PROGRAMMA

Vi mostriamo ora un semplice programma che chiarisce l'uso dell'istruzione OUT.

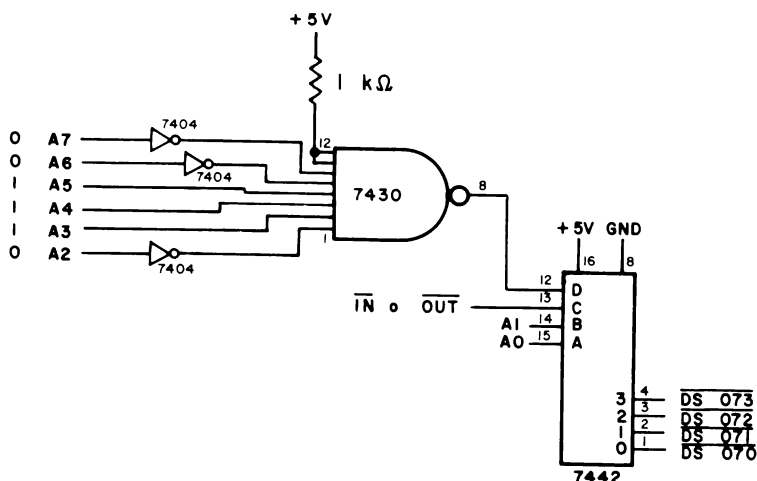



Fig. 4-8. Circuito di decodifica in assoluto basato sull'uso della porta NOR a 8 ingressi 7430 e di un decodificatore 7442.

Indirizzo di memoria LO	*Byte di istruzione ottale*	*Codice mnemonico*	*Descrizione*
000	076	MVI A	Trasferisci il byte seguente nell'accumulatore
001	023	023	Byte di dati
002	323	OUT	Genera un impulso di selezione dispositivo per mettere in uscita otto bit dei dati dell'accumulatore al dispositivo con il codice dato nel byte seguente
003	321	321	Codice per il dispositivo di uscita
004	303	JMP	Salto incondizionato alla posizione di memoria data nei due byte seguenti
005	000		Byte d'indirizzo di memoria LO
006	000		Byte d'indirizzo di memoria HI

Questo semplice programma mette il byte di dati 023 nell'accumulatore, manda i contenuti dell'accumulatore al dispositivo di uscita 321 e, infine, ritorna all'indirizzo di memoria LO 000, punto nel quale il programma si ripete. L'indirizzo di memoria HI è 000 e tutti i numeri suddetti sono in codice ottale a tre cifre.

Se doveste applicare un oscilloscopio ad un microcomputer 8080 ed osservaste l'esecuzione del programma suddetto, osservereste i timing mostrati in Fig. 4-9. I contenuti dell'accumulatore vengo-

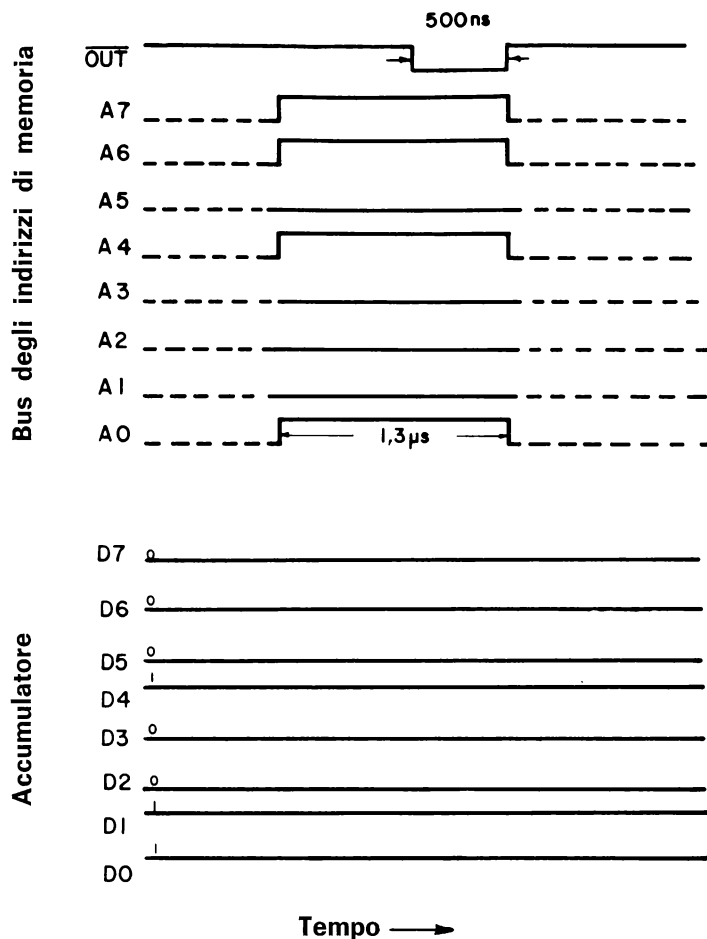



Fig. 4-9. Timing che si può applicare per generare un impulso di selezione dispositivo e per effettuare un latch all'uscita dell'accumulatore durante un'istruzione OUT di 5 µs. Le informazioni vengono trasferite dall'accumulatore ad un dispositivo di uscita e viene fornito un impulso di selezione dispositivo per un periodo di solo 500 ns.

no prima settati e rimangono invariati per tutto il programma: il byte dell'accumulatore ha un valore di 00010011, o 023 in codice ottale. Quando l'istruzione OUT viene eseguita, le linee dell'indirizzo di memoria LO assumono il valore corrispondente al codice dispositivo, che è 321 nel programma suddetto. Questo valore viene mantenuto sulle linee d'indirizzo per 1,3 µs, periodo nel quale viene generato il segnale di controllo $\overline{OUT}$, solitamente per 500 ns in un microcomputer 8080 che opera a 2 MHz. La combinazione del

segnale di controllo $\overline{\text{OUT}}$ e delle otto linee d'indirizzo è sufficiente per generare 256 diversi impulsi di selezione dispositivi, come mostra la Fig. 4-3. I dati nell'accumulatore restano invariati durante tutta l'istruzione $\overline{\text{OUT}}$ di 5 μs. L'impulso di selezione dispositivo generato in questo periodo si può usare per effettuare un latch su questa informazione dell'accumulatore, come spiegheremo nel Capitolo 7.

IMPULSI DI SELEZIONE DISPOSITIVO USATI COME IMPULSI DI CONTROLLO

Gli impulsi di selezione dispositivo non si usano solo per fornire impulsi di strobe per il trasferimento dei dati fra il microcomputer e i dispositivi di ingresso/uscita. Essi possono essere usati come impulsi di controllo per attivare o far fermare macchine, per azzerare dispositivi, per aprire valvole, ecc. Il circuito di Fig. 4-10 mostra come un impulso di selezione dispositivo può essere usato per azzerare il circuito di un contatore a decade 7490. In qualunque momento vogliate azzerare il contatore, inserite una istruzione OUT nel programma con l'indirizzo del dispositivo appropriato. L'impulso di selezione dispositivo specifico usato è ottenuto dallo schema di decodifica.

Un impulso di selezione dispositivo si può usare insieme ad un latch, come il chip 74175 mostrato nella Fig. 4-11, per memorizzare i dati degli switch, che in questo caso sono visualizzati su di un display a sette segmenti. La memorizzazione dei dati avviene solo quando il 74175 è sottoposto a clock. Il latch può essere azzerato da un altro impulso di selezione dispositivo applicato all'ingresso di clear del chip. Il tempo necessario per azzerare il latch o per memorizzare i dati degli switch, è determinato dal programma del vostro microcomputer. Per esempio, inizialmente potreste voler sottoporre a clock il latch ed osservare il set dei vari switch logici sul display. Potete farlo con l'aiuto di un impulso di selezione dispositivo di uscita 001, applicato al pin 1 sul chip. Più avanti nel programma, potreste voler azzerare il latch; in questo caso, il compito viene attuato da un impulso di selezione dispositivo di uscita 000. Ecco il tipo di programma che dovreste usare:

Indirizzo di memoria LO	*Byte di istruzione ottale*	*Codice mnemonico*	*Descrizione*
000	323	OUT	Invia un impulso di selezione dispositivo 001 per azzerare il latch
001	001	001	
.	.	.	
.	.	.	
020	323	OUT	
021	000	000	Invia un impulso di selezione dispositivo 000 per visualizzare i dati degli switch logici aggiornati
.	.	.	
.	.	.	

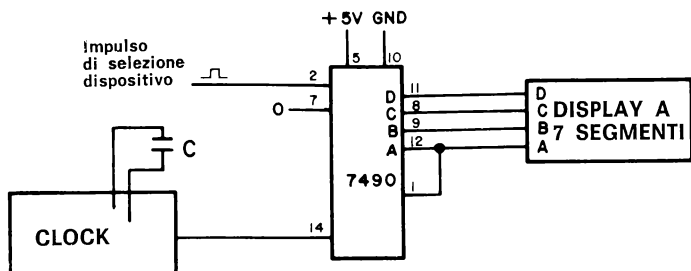



Fig. 4-10. Un'impulso di selezione dispositivo può essere usato per azzerare un contatore, come mostrato per il chip 7490.

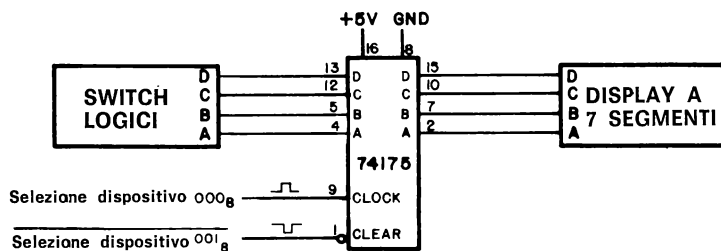



Fig. 4-11. Una coppia di impulsi di selezione dispositivo viene usata per sottoporre a clock un latch e per azzerarlo.

Notate che le istruzioni di uscita sono state usate parecchio per generare gli impulsi di controllo che sono serviti a controllare le operazioni dei chip 7490 e 74175. Si sarebbe potuto usare qualunque impulso di selezione dispositivo con buoni risultati, ma c'è una ragione importante per cui ciò non è stato fatto. In qualunque istante poniate in uscita i dati con un'istruzione OUT, i dati nell'accumulatore vengono copiati in un dispositivo di uscita esterno. Il microcomputer non si preoccupa del fatto che esista o meno un dispositivo che accetti i dati; i contenuti dell'accumulatore non cambiano. Con un'istruzione IN, la situazione è diversa. In qualunque momento venga generato un impulso di selezione dispositivo d'ingresso, il microcomputer si aspetta che otto bit di dati vengano trasferiti all'accumulatore da un dispositivo d'ingresso. Se tale dispositivo non c'è, il registro dell'accumulatore viene di solito caricato con livelli logici tutti 1, cioè con il byte di dati ottale 377. Perciò, dovreste usare sempre istruzioni OUT per generare impulsi di controllo; così facendo, continuate ad avere il controllo sui contenuti dell'accumulatore.

ESEMPIO

Scopo

Lo scopo di questo esempio è mostrare come si possono con-

figurare due decodificatori 74154 per generare sedici impulsi di selezione dispositivi contigui all'interno dei 256 impulsi di selezione dispositivi possibili.

Programma

Indirizzo di memoria LO	*Byte di istruzione ottale*	*Codice mnemonico*	*Descrizione*
000	323	OUT	Invia un impulso di selezione dispositivo al dispositivo con il codice dato dal byte seguente
001	000	000	Codice dispositivo 000_8

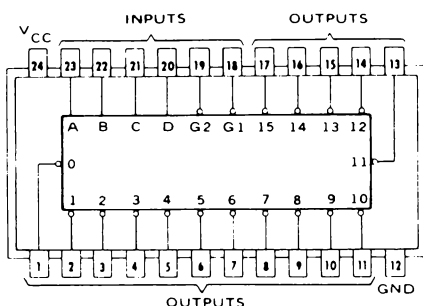
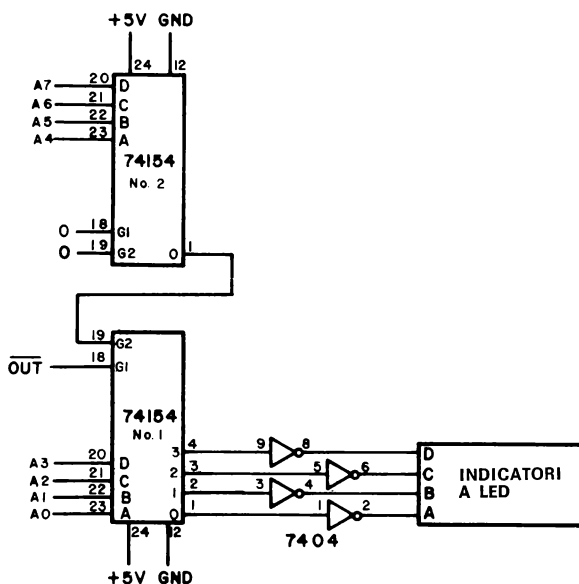
(A) Configurazione dei pin del chip 7404

(B) Configurazione dei pin del chip 74154

(C) Schema del circuito

Fig. 4-12. Circuito per generare sedici impulsi di selezione dispositivo contigui.

002	323	OUT	Invia un impulso di selezione dispositivo al dispositivo con il codice dato dal byte seguente
003	001	001	Codice dispositivo 001_8
004	323	OUT	Invia un impulso di selezione dispositivo al dispositivo con il codice dato dal byte seguente
005	002	002	Codice dispositivo 002_8
006	323	OUT	Invia un impulso di selezione dispositivo al dispositivo con il codice dato dal byte seguente
007	003	003	Codice dispositivo 003_8
010	303	JMP	Salto incondizionato all'indirizzo di memoria dato nei due byte seguenti
011	000		Byte d'indirizzo di memoria LO
012	000		Byte d'indirizzo di memoria HI

Commenti

Eseguendo questo programma, il microcomputer genera impulsi di sincronizzazione $\overline{OUT}$ e fornisce il codice dispositivo sulle linee d'indirizzo LO, da A0 ad A7. Questa informazione viene usata dai decodificatori per produrre un unico impulso, uno su 256, per ogni codice dispositivo. Sebbene solo quattro dei codici possibili siano mostrati nel grafico e generati dal programma, potete generare tutti e sedici gli impulsi aggiungendo software.

Il decodificatore n. 1 74154 ha già tutti i primi sedici canali di uscita decodificati disponibili.

Il decodificatore 74154 n. 2 agisce come decodificatore principale per abilitare, o attivare, il decodificatore 74154 più basso, solo quando le linee d'indirizzo A4, A5, A6 e A7 sono tutte a livello logico 0. I canali di uscita rimanenti sul decodificatore n. 2 si possono usare per abilitare più decodificatori per produrre impulsi di selezione dispositivo di uscita addizionali. Il software suddetto contiene un loop che fa sì che il microcomputer metta continuamente in uscita i quattro impulsi di selezione dispositivo di uscita.

TEST

Questo test prova quanto avete capito circa i concetti e gli esperimenti di questo capitolo. Per favore scrivete le vostre risposte su di un foglio di carta a parte.

4-1. Che cos'è un impulso di selezione dispositivo, e come viene prodotto con l'aiuto di un insieme di circuiti che interfacciano il microcomputer 8080? Usate schemi per rispondere a questa domanda.

4-2. Descrivete dieci diversi utilizzi degli impulsi di selezione dispositivo. Potete usare circuiti integrati diversi nella vostra risposta, ma per favore non ripetetevi.

4-3. Scrivete un semplice programma che generi tre diversi impulsi di selezione dispositivo di ingresso e due diversi impulsi di selezione dispositivo di uscita.

4-4. Disegnate una serie di diagrammi che mostrino in che modo si usa un impulso di selezione dispositivo 164_8 per permettere ad un dispositivo di uscita di effettuare un latch sui contenuti dell'accumulatore, che contiene il byte dati 11010110_2.

Il vostro compito sarà accettabile se sarete stati in grado di rispondere a tutte e quattro queste domande in modo corretto, in 40 minuti ed a libro chiuso. Troverete la maggior parte delle risposte nei capitoli successivi di questo libro.

CHE COSA AVETE REALIZZATO IN QUESTO CAPITOLO?

All'inizio di questo capitolo, era stato stabilito che, alla fine, sareste stati in grado di:

- Identificare le istruzioni OUT e IN in un programma del microcomputer 8080.

  I codici operativi per queste due istruzioni sono **323** *e* **333**, *rispettivamente. In qualunque momento essi appaiano in un programma del microcomputer, essi sono le istruzioni OUT e IN, senza eccezioni.*

- Disegnare lo schema di un circuito che può generare fino a 256 diversi impulsi di selezione dispositivo.

  Questo circuito è stato mostrato nella Fig. 4-3. Esso impiega diciassette circuiti integrati 74154 da quattro a sedici linee.

- Spiegare come vengono generati gli impulsi di selezione dispositivo dal microcomputer 8080.

  Questo capitolo ne ha fornito solo una spiegazione parziale. Ogni impulso di selezione dispositivo viene prodotto come conseguenza di un impulso di strobe $\overline{OUT}$ o $\overline{IN}$ e di un impulso prodotto dalla decodificazione del codice dispositivo a 8 bit che appare per 1,3 μs sul bus degli indirizzi di memoria. Non abbiamo parlato di come vengono prodotti gli impulsi $\overline{OUT}$ e $\overline{IN}$. E' un argomento che tratteremo nel Capitolo 6.

- Scrivere dei semplici programmi che usino le istruzioni IN o OUT.

  Questo è stato fatto in molti esempi nel nostro capitolo. La maggior parte dei programmi contengono solo cinque byte istruzioni.

- Disegnare lo schema a blocchi di un decodificatore 74154 da quattro a sedici linee.

  Abbiamo usato il chip 74154 molte volte in questo capitolo, perciò dovreste essere in grado di disegnarne uno schema a blocchi e di identificare a memoria alcuni dei pin più importanti. G1 e G2 sono ai pin 18 e 19, rispettivamente. I canali di uscita iniziano al pin 1 e continuano fino al pin 11. Riprendono al pin 13.

CAPITOLO 5

Cicli di Clock e Loop di Timing

In questo capitolo, esaminerete il timing di varie istruzioni dell'8080 e l'utilizzazione del microcomputer come generatore di impulsi, temporizzati in modo opportuno. I microcomputer sono usati, frequentemente, in situazioni in cui sono richiesti complessi periodi di timing e particolari sequenze. Ad esempio nel caso del controllo dei semafori, del range di lavoro dei forni a microonde, nel controllo delle lavatrici domestiche.

OBIETTIVI

Alla fine di questo capitolo, sarete in grado di:

- Definire i termini: loop, loop di timing, periodo, clock, ciclo di clock, stato.
- Programmare un loop di timing che può generare delay multipli di 0,200 s.
- Programmare un loop di timing che può generare delay multipli di circa 0,5 ms.
- Dimostrare come un microcomputer può agire come multivibratore monostabile.

DEFINIZIONI

Clock cycle (Ciclo di clock) Un singolo periodo di clock.

Loop: *(Lap)*	Una sequenza di istruzioni che è eseguita ripetutamente, finché non si raggiune una condizione particolare di fine.[3]
Monostable multivibrator (Multivibratore monostabile)	Un circuito che ha un solo stato stabile, da cui può essere spostato, ma solo per un predeterminato intervallo, dopodiché ritorna allo stato originale.
Period (Periodo)	Il tempo richiesto per un ciclo completo di una serie regolare e ripetitiva di eventi.
Programmable sequencer (Sequenziatore programmabile)	Un sequenziatore in cui l'ordine di successione degli eventi può essere cambiato con l'ausilio della programmazione.
Sequencer (Sequenziatore)	Un dispositivo elettronico che può essere settato per iniziare una serie di eventi, in modo da assicurare anche una data sequenza agli eventi stessi.[4]
State (Stato)	Un singolo periodo di clock o condizione stabile.
Timing loop (Loop di timing)	Un loop che richiede, per la sua esecuzione, un preciso periodo di tempo.

MULTIVIBRATORI MONOSTABILI

Un multivibratore monostabile può essere definito come segue:

Monostable multivibrator (Multivibratore monostabile)	Un circuito che ha un solo stato stabile, da cui può essere spostato, ma solo per un predeterminato intervallo, dopodiché ritorna allo stato originale.

Sono usati per generare singoli impulsi di clock di durata nota. Tali impulsi sono ampiamente usati in elettronica digitale per coordinare i vari timing dei circuiti digitali ed anche per regolare l'ON/OFF di dispositivi esterni in predeterminati intervalli di tempo. Gli schemi di Fig. 5-1 indicano i più comuni multivibratori attualmente utilizzati. I chip 7412, 74122 e 74123 generano singoli impulsi di clock la cui ampiezza varia da circa 40 ns ad alcuni millisecondi. Il monostabile 555 può generare impulsi di ampiezza variabile dai microsecondi ai minuti. Questi chip sono stati presentati in modo molto dettagliato nel capitolo 8 del *Bugbook II.*

IL MICROCOMPUTER COME MULTIVIBRATORE MONOSTABILE

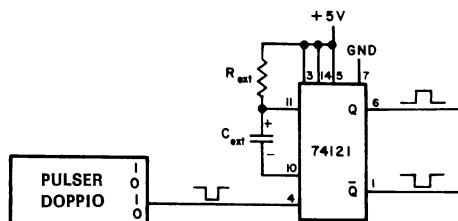
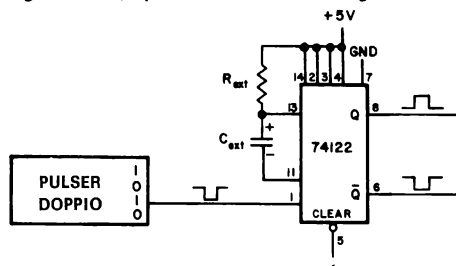
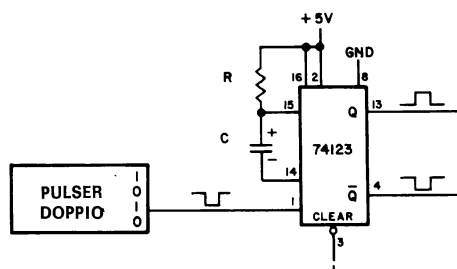
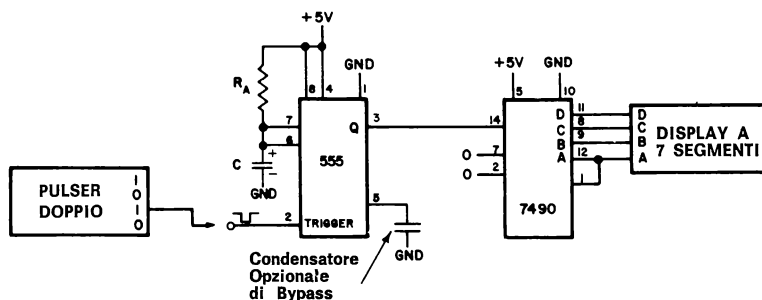
Un microcomputer, come l'8080, può anche funzionare come multivibratore. E' possibile scrivere un programma che contiene un *loop di timing:*

(A) Chip 74121 usato come multivibratore monostabile. Possono facilmente essere generati impulsi dell'ordine di grandezza di 40 ns

(B) Chip 74122 nella configurazione di multivibratore monostabile.

(C) Chip 74123 nella configurazione di multivibratore monostabile. Sono disponibili due monostabili indipendenti sul chip 74123.

(D) Il timer 555 usato come multivibratore monostabile. Non è possibile avere impulsi più brevi di alcuni microsecondi. L'impulso di strobe deve essere più breve dell'impulso in uscita.

Fig. 5-1. Alcuni comuni multivibratori monostabili.

Timing loop — Un loop che richiede, per la sua esecuzione, un preciso periodo di tempo.

Loop — Una sequenza di istruzioni che è eseguita ripetutamente, finché non si raggiunge una condizioni particolare di fine.[3]

Dovrebbe essere chiaro che il loop deve essere posto tra due istruzioni di OUT. Con una opportuna circuiteria esterna, un singolo impulso di clock di data lunghezza, può essere facilmente generato dal software. Possiamo chiamare tale impulso come «impulso monostabile generato da software».

La larghezza dell'impulso, cioè la sua durata, può essere controllata modificando il software piuttosto che resistenze o condensatori, come nel caso in cui vengono utilizzati i chip 74121, 74122, 74123 oppure il 555.

Questo ci porta a fare la seguente domanda: Come è possibile sapere quanto tempo ci vuole per eseguire un dato programma o sequenza di istruzioni? Sarà questo l'oggetto di quanto segue.

QUANTO CI VUOLE PER ESEGUIRE UNA ISTRUZIONE?

Un microcomputer, come ogni qualsiasi computer digitale, è un dispositivo elettronico digitale «clocked», cioè le sue operazioni si realizzano durante intervalli di clock. Ad esempio un tipico microprocessore 8080 opera con un clock di 2 MHz. Un singolo ciclo di clock, o *stato,* ha un periodo di:

periodo = 1 ciclo/2.000.000 cicli al secondo (Hz)
0,0000005 s
0,5 μs
500 ns

dove per *ciclo di clock, stato,* e *periodo* si intende:

Ciclo di clock Un singolo periodo di clock.
Stato Un singolo periodo di clock o condizione stabile.
Periodo Il tempo richiesto per un ciclo completo di una serie regolare e ripetitiva di eventi.

Ogni azione entro il microprocessore 8080 richiede alcuni multipli del periodo di clock del microprocessore. Una istruzione è una tipica «azione» del microprocessore, ed è realizzata in multipli del periodo di clock del microprocesore. L'istruzione più veloce richiede solo quattro cicli di clock, detti stati:

$$4 \text{ cicli} = 4 \times 500 \text{ ns} = 2\ \mu\text{s}$$

mentre la più lenta istruzione, XTHL, richiede diciotto cicli di clock:

$$18 \text{ cicli} = 18 \times 500 \text{ ns} = 9\ \mu\text{s}$$

Quindi, il tempo richiesto per eseguire un gruppo di istruzioni è determinato dal tempo totale, somma dei tempi necessari alla esecuzione delle singole istruzioni del gruppo moltiplicando per il numero delle volte per cui le istruzioni stesse sono eseguite. Facendo attenzione, è possibile definire un gruppo di istruzioni il cui tempo di esecuzione è ben preciso, ad esempio 0.20 s. Al rimanere del clock rate al valore 2 MHz ci vorranno sempre 0,2 s per eseguire quel gruppo di istruzioni.

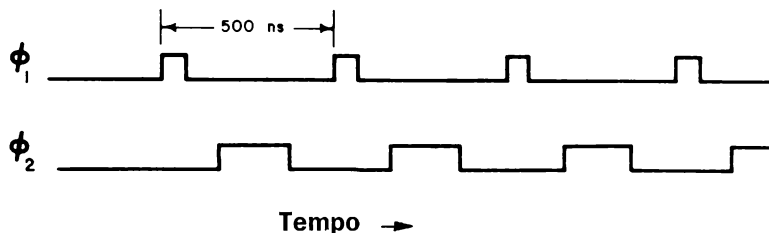



Fig. 5-2. Diagramma dei tempi per gli ingressi di clock a due fasi del microprocessore 8080.

In generale, il microcomputer si valuta secondo la velocità; più velocemente esegue una serie di istruzioni, più potente è ed ancora più utilmente si configura nella sostituzione della logica cablata, come i chip della serie 7400. Attualmente, una frequenza tipica per l'8080 è 2 MHz; ciò significa che un singolo periodo di clock è eguale a 500 ns. Questo è la durata tra due successivi impulsi di clock $\varnothing_1$ e $\varnothing_2$, come in Fig. 5-2.

Tenete ben presente che la velocità dei chip 8080 può essere aumentata tramite miglioramenti tecnologici. E' praticamente possibile che un 8080 «improved» possa operare a 4 MHz, cioè con un periodo di clock di 250 ns. Naturalmente, le memorie R/W, PROM e ROM dovranno essere le più veloci sul mercato, per adattarsi alla rapidità operativa generale. L'8080, inoltre, non è certo l'ultima parola nel campo dei microprocessori. Sviluppi nella tecnologia dei semiconduttori, nell'area delle *tecniche bipolari,* permetteranno la nascita di microprocessori con clock rate di 30 MHz, o 33,3 ns.

LISTING DEI CICLI PER LE ISTRUZIONI COSTITUENTI IL SET DELL'8080

Nella pagina seguente è presentato un listing completo del set di istruzioni dell'8080, con il numero dei cicli di clock necessari per ogni istruzioni. Questa pagina è ricavata dalla documentazione della Intel Corporation. In questa sezione, faremo un riassunto delle condizioni che si possono dedurre dal listing.

Il tempo più breve durante il quale può essere eseguita una

INSTRUCTION SET

Summary of Processor Instructions

Mnemonic	Description	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock[2] Cycles
		Instruction Code[1]								
MOV r1, r2	Move register to register	0	1	0	0	0	S	S	S	5
MOV M, r	Move register to memory	0	1	1	1	0	S	S	S	7
MOV r, M	Move memory to register	0	1	0	0	0	1	1	0	7
HLT	Halt	0	1	1	1	0	1	1	0	7
MVI r	Move immediate register	0	0	0	0	0	1	1	0	7
MVI M	Move immediate memory	0	0	1	1	0	1	1	0	10
INR r	Increment register	0	0	0	0	0	1	0	0	5
DCR r	Decrement register	0	0	0	0	0	1	0	1	5
INR M	Increment memory	0	0	1	1	0	1	0	0	10
DCR M	Decrement memory	0	0	1	1	0	1	0	1	10
ADD r	Add register to A	1	0	0	0	0	S	S	S	4
ADC r	Add register to A with carry	1	0	0	0	1	S	S	S	4
SUB r	Subtract register from A	1	0	0	1	0	S	S	S	4
SBB r	Subtract register from A with borrow	1	0	0	1	1	S	S	S	4
ANA r	And register with A	1	0	1	0	0	S	S	S	4
XRA r	Exclusive Or register with A	1	0	1	0	1	S	S	S	4
ORA r	Or register with A	1	0	1	1	0	S	S	S	4
CMP r	Compare register with A	1	0	1	1	1	S	S	S	4
ADD M	Add memory to A	1	0	0	0	0	1	1	0	7
ADC M	Add memory to A with carry	1	0	0	0	1	1	1	0	7
SUB M	Subtract memory from A	1	0	0	1	0	1	1	0	7
SBB M	Subtract memory from A with borrow	1	0	0	1	1	1	1	0	7
ANA M	And memory with A	1	0	1	0	0	1	1	0	7
XRA M	Exclusive Or memory with A	1	0	1	0	1	1	1	0	7
ORA M	Or memory with A	1	0	1	1	0	1	1	0	7
CMP M	Compare memory with A	1	0	1	1	1	1	1	0	7
ADI	Add immediate to A	1	1	0	0	0	1	1	0	7
ACI	Add immediate to A with carry	1	1	0	0	1	1	1	0	7
SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	7
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	7
ANI	And immediate with A	1	1	1	0	0	1	1	0	7
XRI	Exclusive Or immediate with A	1	1	1	0	1	1	1	0	7
ORI	Or immediate with A	1	1	1	1	0	1	1	0	7
CPI	Compare immediate with A	1	1	1	1	1	1	1	0	7
RLC	Rotate A left	0	0	0	0	0	1	1	1	4
RRC	Rotate A right	0	0	0	0	1	1	1	1	4
RAL	Rotate A left through carry	0	0	0	1	0	1	1	1	4
RAR	Rotate A right through carry	0	0	0	1	1	1	1	1	4
JMP	Jump unconditional	1	1	0	0	0	0	1	1	10
JC	Jump on carry	1	1	0	1	1	0	1	0	10
JNC	Jump on no carry	1	1	0	1	0	0	1	0	10
JZ	Jump on zero	1	1	0	0	1	0	1	0	10
JNZ	Jump on no zero	1	1	0	0	0	0	1	0	10
JP	Jump on positive	1	1	1	1	0	0	1	0	10
JM	Jump on minus	1	1	1	1	1	0	1	0	10
JPE	Jump on parity even	1	1	1	0	1	0	1	0	10
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	10
CALL	Call unconditional	1	1	0	0	1	1	0	1	17
CC	Call on carry	1	1	0	1	1	1	0	0	11/17
CNC	Call on no carry	1	1	0	1	0	1	0	0	11/17
CZ	Call on zero	1	1	0	0	1	1	0	0	11/17
CNZ	Call on no zero	1	1	0	0	0	1	0	0	11/17
CP	Call on positive	1	1	1	1	0	1	0	0	11/17
CM	Call on minus	1	1	1	1	1	1	0	0	11/17
CPE	Call on parity even	1	1	1	0	1	1	0	0	11/17
CPO	Call on parity odd	1	1	1	0	0	1	0	0	11/17
RET	Return	1	1	0	0	1	0	0	1	10
RC	Return on carry	1	1	0	1	1	0	0	0	5/11
RNC	Return on no carry	1	1	0	1	0	0	0	0	5/11
RZ	Return on zero	1	1	0	0	1	0	0	0	5/11
RNZ	Return on no zero	1	1	0	0	0	0	0	0	5/11
RP	Return on positive	1	1	1	1	0	0	0	0	5/11
RM	Return on minus	1	1	1	1	1	0	0	0	5/11
RPE	Return on parity even	1	1	1	0	1	0	0	0	5/11
RPO	Return on parity odd	1	1	1	0	0	0	0	0	5/11
RST	Restart	1	1	A	A	A	1	1	1	11
IN	Input	1	1	0	1	1	0	1	1	10
OUT	Output	1	1	0	1	0	0	1	1	10
LXI B	Load immediate register Pair B & C	0	0	0	0	0	0	0	1	10
LXI D	Load immediate register Pair D & E	0	0	0	1	0	0	0	1	10
LXI H	Load immediate register Pair H & L	0	0	1	0	0	0	0	1	10
LXI SP	Load immediate stack pointer	0	0	1	1	0	0	0	1	10
PUSH B	Push register Pair B & C on stack	1	1	0	0	0	1	0	1	11
PUSH D	Push register Pair D & E on stack	1	1	0	1	0	1	0	1	11
PUSH H	Push register Pair H & L on stack	1	1	1	0	0	1	0	1	11
PUSH PSW	Push A and Flags on stack	1	1	1	1	0	1	0	1	11
POP B	Pop register pair B & C off stack	1	1	0	0	0	0	0	1	10
POP D	Pop register pair D & E off stack	1	1	0	1	0	0	0	1	10
POP H	Pop register pair H & L off stack	1	1	1	0	0	0	0	1	10
POP PSW	Pop A and Flags off stack	1	1	1	1	0	0	0	1	10
STA	Store A direct	0	0	1	1	0	0	1	0	13
LDA	Load A direct	0	0	1	1	1	0	1	0	13
XCHG	Exchange D & E, H & L Registers	1	1	1	0	1	0	1	1	4
XTHL	Exchange top of stack, H & L	1	1	1	0	0	0	1	1	18
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1	5
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	5
DAD B	Add B & C to H & L	0	0	0	0	1	0	0	1	10
DAD D	Add D & E to H & L	0	0	0	1	1	0	0	1	10
DAD H	Add H & L to H & L	0	0	1	0	1	0	0	1	10
DAD SP	Add stack pointer to H & L	0	0	1	1	1	0	0	1	10
STAX B	Store A indirect	0	0	0	0	0	0	1	0	7
STAX D	Store A indirect	0	0	0	1	0	0	1	0	7
LDAX B	Load A indirect	0	0	0	0	1	0	1	0	7
LDAX D	Load A indirect	0	0	0	1	1	0	1	0	7
INX B	Increment B & C registers	0	0	0	0	0	0	1	1	5
INX D	Increment D & E registers	0	0	0	1	0	0	1	1	5
INX H	Increment H & L registers	0	0	1	0	0	0	1	1	5
INX SP	Increment stack pointer	0	0	1	1	0	0	1	1	5
DCX B	Decrement B & C	0	0	0	0	1	0	1	1	5
DCX D	Decrement D & E	0	0	0	1	1	0	1	1	5
DCX H	Decrement H & L	0	0	1	0	1	0	1	1	5
DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1	5
CMA	Complement A	0	0	1	0	1	1	1	1	4
STC	Set carry	0	0	1	1	0	1	1	1	4
CMC	Complement carry	0	0	1	1	1	1	1	1	4
DAA	Decimal adjust A	0	0	1	0	0	1	1	1	4
SHLD	Store H & L direct	0	0	1	0	0	0	1	0	16
LHLD	Load H & L direct	0	0	1	0	1	0	1	0	16
EI	Enable Interrupts	1	1	1	1	1	0	1	1	4
DI	Disable interrupt	1	1	1	1	0	0	1	1	4
NOP	No-operation	0	0	0	0	0	0	0	0	4

NOTE: 1. DDD o SSS - 000 B - 001 C - 010 D - 011 E - 100 H - 101 L - 110 Memory - 111 A

2. Due possibili tempi di ciclo, (5/11) indicano che il ciclo dell'istruzione dipende dai flag di condizione

istruzione, se si ipotizza che un singolo ciclo di clock richiede 500 ns, è 4 cicli di clock, o 2 μs. Le operazioni logiche ed aritmetiche sui registri e le rotazioni sui contenuti dell'accumulatore possono essere eseguite in questo breve intervallo di tempo. In questo gruppo vi sono le istruzioni ADD, ADC, SUB, SBB, ANA, XRA, ORA, CMP, RLC, RRC, RAL e RAR.

Tutte le istruzioni di MOV da registro a registro, come pure quella di incremento e decremento come INR, DCR, INX e DCX, sono eseguite in cinque cicli di clock, o 2.5 μs.

La maggior parte delle istruzioni in cui sono coinvolti trasferimenti di informazioni verso o dalla memoria, richiedono almeno sette cicli di clock, o 3.5 μs. In questi gruppi sono comprese le istruzioni registro-memoria o memoria-registro MOV, le istruzioni ADD M, ADC M, SUB M, ANA M, XRA M, ORA M, CMP M, MVI, ADI, ACI, SUI, SBI, ANI, XRI, ORI, CPI, ed anche STAX e LDAX. Tutte le istruzioni di salto (Jump) condizionato ed incondizionato: MVI M, INR M, DCR M, RET, IN, OUT, LXI, DAD e POP, richiedono dieci cicli o 5 μs per la loro esecuzione.

Le istruzioni di «call» condizionato richiedono undici o diciassette cicli di clock, in funzione del fatto che la subroutine sia chiamata o meno. Se è chiamata, sono necessari diciassette cicli di clock, o 8.5 μs. In caso contrario servono solo undici cicli, o 5.5 μs.

Stesse condizioni sono applicabili per le istruzioni di ritorno condizionato. Se il flag è in una condizione tale per cui si realizza un ritorno, servono undici cicli, cioè 5.5 μs.; se non c'è ritorno, cinque cicli, cioè 2,5 μs. Se volete studiare e realizzare un loop di timing molto preciso, vi serve questo listing. Potrebbe risultarvi utile fare una fotocopia da utilizzare in sede di programmazione. Questo listing è incluso nell'*Intel 8080 Microcomputer System Manual.*

IL CONTEGGIO DEI CICLI DI CLOCK: ALCUNI SEMPLICI ESEMPI DI PROGRAMMI PER MICROCOMPUTER

In questa sezione, prenderemo in considerazione alcuni semplici programmi ed impareremo come conteggiare i cicli di clock.

Programma N. 1

Indirizzo di memoria LO	*Byte di istruzione ottale*	*Codice mnemonico*	*Descrizione*
000	303	JMP	Salto incondizionato all'indirizzo di memoria dato nei due byte seguenti
001	000		Byte d'indirizzo LO
002	000		Byte d'indirizzo HI

Se guardate l'elenco del set di istruzioni della Intel, osserverete che un'istruzione JMP richiede dieci cicli di clock per essere ese-

guita. Dato che questo programma non contiene nessuna istruzione IN o OUT, incontrerete delle difficoltà nel misurare, o nell'impiegare questo programma per generare impulsi monostabili di 5 μs. Questo programma può essere scartato per questo scopo. E' importante notare che tutti e tre i byte dell'istruzione JMP vengono «usati» in dieci cicli di clock. Questo tempo complessivo si applica a tutti i byte in un'istruzione a più byte.

Programma N. 2

Indirizzo di memoria LO	*Byte di istruzione ottale*	*Codice mnemonico*	*Cicli di clock*	*Descrizione*
000	323	OUT	10	Genera un impulso di selezione dispositivo per il dispositivo che ha il codice seguente
001	000	000		Codice dispositivo
002	323	OUT	10	Genera il secondo impulso di selezione dispositivo per lo stesso dispositivo
003	000	000		Codice dispositivo
004	166	HLT	7	Alt

Questo programma viene eseguito in 27 cicli di clock, o 13,5 μs. Quello che ci interessa di questo programma, comunque, è che vengono generati e mandati allo stesso dispositivo due impulsi di selezione dispositivo. Un impulso viene usato per attivare il dispositivo e l'altro per disattivarlo. Quale dispositivo può venire attivato e disattivato da un solo ingresso? Come esempio, considerare il circuito flip-flop J-K «toggled» della Fig. 5-3. Gli ingressi di preset, di clear (azzeramento), J e K, vengono lasciati tutti scollegati, il che significa che sono tutti al livello logico 0.

La prima istruzione OUT nel Programma n. 2 genera un solo impulso di selezione dispositivo che cambia il livello logico del flip-flop in 1. Questa uscita viene usata come un segnale di gating per la porta AND a due ingressi 7408, e permette ai segnali provenienti dal clock $\varnothing_2$ di passare attraverso la porta fino al conta-

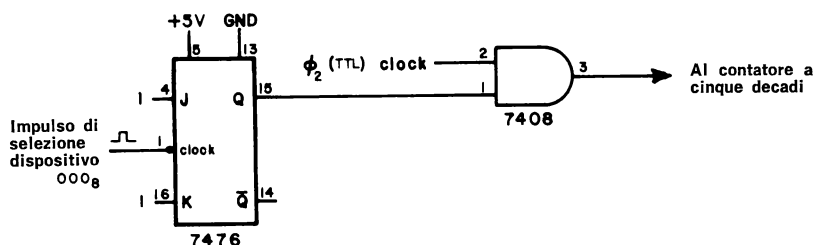



Fig. 5-3. Un flip-flop 7476 toggled (commutabile) all'ingresso di clock, può essere usato per effettuare un gate sugli impulsi di clock del clock principale in un microprocessore basato sull'8080.

tore a cinque decadi, dove vengono conteggiati. La seconda istruzione OUT del Programma n. 2 genera un impulso di selezione dispositivo che fa sì che l'uscita Q al pin 15 ritorni a livello logico 0; questa azione disabilita la porta 7408 e arresta il conteggio da parte del contatore a cinque decadi.

Per quanto tempo «conterà» il contatore? Si può rispondere meglio a questa domanda con un'informazione che sarà fornita nel Capitolo 6. Possiamo anticipare questa informazione rilevando che l'impulso di selezione dispositivo non viene mandato al flip-flop 7476 fino agli ultimi 500 ns dell'istruzione OUT di 5 μs. Si può far vedere che il flip-flop è ad uno stato logico 1 solo per dieci cicli di clock, o 5 μs: 500 ns durante la prima istruzione OUT e 4,5 μs durante la seconda istruzione OUT. L'intervallo di tempo di 5 μs è del tutto esatto: non sono 4,93 o 5,05 μs, ma precisamente 5 μs per un ingresso di clock 8080 a 2 MHz controllato da un quarzo.

Programma N. 3

Indirizzo di memoria LO	*Byte di istruzione ottale*	*Codice mnemonico*	*Cicli di clock*	*Descrizione*
000	323	OUT	10	Genera l'impulso di selezione dispositivo
001	000	000		Codice dispositivo
002	*	*	*	Istruzione aritmetica, logica o altra ad un solo byte
003	323	OUT	10	Genera un impulso di selezione dispositivo
004	000	000		Codice dispositivo
005	166	HLT	7	Alt

Il programma n. 3 è simile al programma n. 2, ad eccezione del fatto che abbiamo inserito un'istruzione fra le due istruzioni OUT. Si possono provare tutte le istruzioni ad un solo byte, ad eccezione delle istruzioni di ritorno. Per esempio, se l'istruzione RLC, che richiede quattro cicli di clock, venisse inserita all'indirizzo di memoria 002_8, il tempo totale di conteggio, misurato dal contatore a cinque decadi e dal circuito 7476 mostrato nella Fig. 5-3; sarebbe:

$$\begin{aligned} \text{tempo di conteggio} &= (4 + 10) \times 500 \text{ ns} \\ &= 7000 \text{ ns} \\ &= 7\ \mu\text{s} \end{aligned}$$

Se sottraeste 5 μs, il tempo richiesto per il programma n. 2, da questo valore, misurerebbe un tempo di 2 μs per l'istruzione RLC, un tempo che corrisponde a quattro cicli di clock, come indicato nel diagramma di timing dell'8080.

Programma N. 4

Indirizzo di memoria LO	*Byte di istruzione ottale*	*Codice mnemonico*	*Cicli di clock*	*Descrizione*
000	323	OUT	10	Genera un impulso di selezione dispositivo, che setta il flip-flop 7476
001	000	000		Codice dispositivo
002	323	OUT	10	Genera un impulso di selezione dispositivo che azzera il flip-flop 7476
003	001	001		Codice dispositivo, diverso
004	166	HLT	7	Alt

Invece di cambiare lo stato del flip-flop 7476, potete collegare due diverse linee di segnali di selezione dispositivo agli ingressi di preset e clear del flip-flop. Il circuito è mostrato nella Fig. 5-4. Dovrebbe essere evidente che questo è un modo uguale, o forse più valido di cambiare gli stati di uscita su di un flip-flop. In questo caso, sebbene il flip-flop 7476 sia un solo «dispositivo», sono necessari due impulsi di selezione dispositivo per controllarlo. Non occorre che vi sia nessuna correlazione fra il numero di dispositivi presenti e il numero di impulsi di selezione dispositivo richiesto per servire i dispositivi stessi.

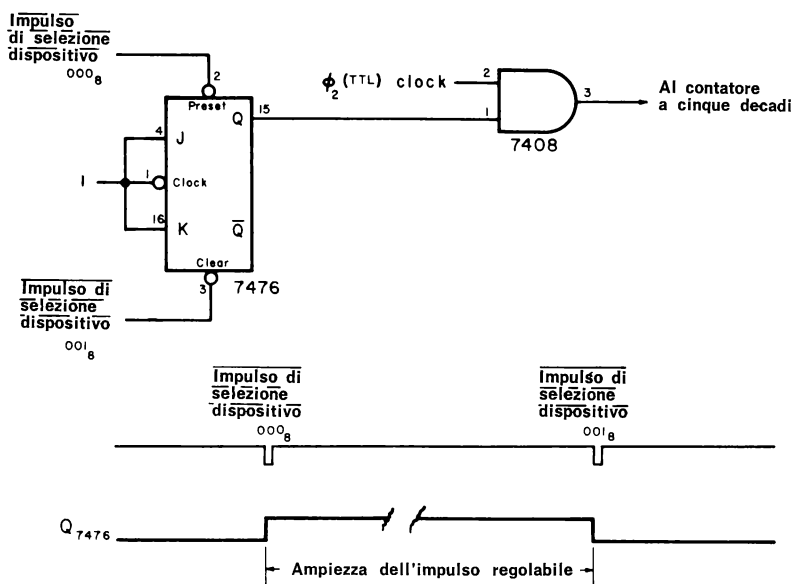



Fig. 5-4. Un semplice circuito flip-flop 7476 e i diagrammi di tempo associati. Questo circuito si può usare per conteggiare i cicli di clock in un microprocessore basato sull'8080.

Programma N. 5

Indirizzo di memoria LO	*Byte di istruzione ottale*	*Codice mnemonico*	*Cicli di clock*	*Descrizione*
000	006	MVI B	7	Trasferisci il byte di dati seguente nel registro B
001	*			Byte di dati
002	323	OUT	10	Genera un impulso di selezione dispositivo che setta il flip-flop 7476
003	000	000		Codice dispositivo
004	005	DCR B	5	Decrementa di uno i contenuti del registro B
005	302	JNZ	10	Salto incondizionato. Se i contenuti del registro B sono 000, ignora questa istruzione e vai all'indirizzo di memoria 010. Altrimenti, salta all'indirizo di memoria dato nei due byte seguenti.
006	004			Byte d'indirizzo LO
007	000			Byte d'indirizzo HI
010	323	OUT	10	Genera un impulso di selezione dispositivo che azzera il flip-flop 7476
011	001	001		Codice dispositivo diverso
012	166	HLT	7	Alt

Questo interessante programma contiene:

- Un'istruzione di salto condizionato (all'indirizzo di memoria 005)
- Un loop di timing (agli indirizzi di memoria da 004 a 007)
- Un byte di dati che determina la durata di tempo del loop di temporizzazione (all'indirizzo di memoria 001)
- Due istruzioni OUT che generano due diversi impulsi di selezione dispositivo.

Il circuito che accompagna questo programma, insieme ad una coppia di diagrammi di timing, è mostrato nella Fig. 5-4. La prima istruzione OUT manda un impulso di selezione dispositivo all'ingresso di preset del flip-flop 7476 al pin 2. L'uscita del flip-flop, Q, viene così settata a livello logico 1, aprendo la porta 7408. L'istruzione OUT che appare all'indirizzo di memoria 010_8 del programma azzera il flip-flop e chiude la porta. L'istruzione di salto condizionato, JNZ, all'indirizzo di memoria 005, permette un salto solo quando il flag di zero è a livello logico 0. L'istruzione di decremento all'indirizzo di memoria 004 diminuirà i contenuti del registro B di uno in ogni passaggio attraverso il loop. Eventualmente, il registro B diventa uguale a 000, momento in cui il controllo di programma si sposta all'istruzione che si trova all'indirizzo di memoria 010, in cui si trova l'istruzione JNZ. La lunghezza del byte di dati all'indirizzo di memoria 001 determina il

numero di loop che avvengono durante l'esecuzione del programma. Se questo byte di dati è **001**, il programma effettuerà un solo passaggio attraverso le istruzioni che vanno dall'indirizzo di memoria 004 all'indirizzo 007. D'altra parte, se il byte di dati è inizialmente **000**. il programma effettuerà un loop 256 volte. Questo avviene perché il byte di dati è decrementato prima dalla prova.

Ogni volta che il programma effettua un loop attraverso gli indirizzi da 004 a 007, c'è un ritardo di quindici cicli clock, o 7,5 μs. Questo tempo è moltiplicato per il numero di passaggi del loop che hanno luogo. A questa figura viene aggiunto 5 μs, il tempo associato con l'azzeramento del flip-flop tramite l'istruzione OUT all'indirizzo di memoria 010.

L'asterisco (*) all'indirizzo di memoria 001 indica che questo byte di dati può variare da **000** a **377**, in codice ottale. Alcuni calcoli dimostreranno come potete variare il tempo in cui viene eseguito il programma n. 5. L'obiettivo è quello di dimostrare, per mezzo di calcoli, *come si possano usare i microcomputer per generare ritardi di tempo in circuiti digitali esterni. Il microcomputer genera questi ritardi usando impulsi di selezione dispositivo e loop di temporizzazione.*

Supponiamo che il byte di dati all'indirizzo di memoria 001 sia uguale a **001**. Questo è il contenuto del registro B alla fine del byte di istruzioni all'indirizzo di memoria 003_8. All'indirizzo di memoria 004, i contenuti del registro B vengono decrementati di 1, e diventano uguali a 000. Con l'aiuto di un'istruzione di salto condizionato, che segue, il controllo di programma viene trasferito all'indirizzo di memoria 010, punto nel quale un'impulso di selezione dispositivo viene generato per azzerare il flip-flop. Il numero totale di cicli di clock fra la prima e la seconda istruzione OUT è:

$$\begin{aligned}\text{numero dei cicli di clock} &= [\text{DCR}] + [\text{JNZ}] + [\text{OUT}]\\ &= \quad 5 \quad + \quad 20 \quad + \quad 10\\ &= 25\end{aligned}$$

A 500 ns per ogni ciclo clock, il tempo totale trascorso è:

$$\begin{aligned}\text{tempo trascorso} &= 25 \times 500 \text{ ns}\\ &= 12{,}5\ \mu\text{s}\end{aligned}$$

Il contatore a cinque decadi del circuito indicato in Fig. 5-4 conta gli impulsi di clock per questo periodo.

Ora supponiamo che il byte di dati all'indirizzo di memoria 001 sia **002** Il numero totale di cicli clock fra la prima e la seconda istruzione OUT è ora:

$$\begin{aligned}\text{numero dei cicli di clock} &= 2\ [\text{DCR}] + 2\ [\text{JNR}] + [\text{OUT}]\\ &= \quad 10 \quad + \quad 20 \quad + \quad 10\\ &= 40\end{aligned}$$

che equivale al seguente tempo totale trascorso di:

tempo trascorso = 40 x 500 ns
= 20 μs

Se il byte di dati all'indirizzo di memoria 011 è *000*, vi sono 256 passi attraverso il loop di temporizzazione. Il numero di cicli clock e il tempo totale trascorso fra le due istruzioni OUT è:

numero dei cicli di clock = 256 [DCR] + 256 [JNZ] + [OUT]
= 1280 + 2560 + 10
= 3850

tempo trascorso = 3850 x 500 ns
= 1925 μs
= 1,925 ms

Questo tempo trascorso di 1.925 ms si riferisce al periodo in cui l'uscita del flip-flop 7476 è in uno stato di livello logico 1, non al tempo totale del programma che è trascorso.

Dovrebbe essere chiaro che la ampiezza dell'impulso mostrato nella Fig. 5-4 è regolabile. L'ampiezza esatta è determinata dalla posizione della prima e della seconda istruzione OUT (quelle che generano gli impulsi di selezione dispositivi 000 e 001) nel programma. Per esempio, l'ampiezza dell'impulso avrebbe i seguenti valori man mano che la grandezza del byte di dati presente all'indirizzo di memoria 001 varia da 000 a 377:

Byte di dati e indirizzo di memoria LO 001	*Numero dei cicli di clock*	*Ampiezza dell'impulso*
000	3850	1,925
001	25	0,0125
002	40	0,02
003	55	0,0275
004	70	0,035
005	85	0,0425
010	130	0,065
020	250	0,125
050	610	0,305
100	970	0,485
200	1930	0,965
300	2890	1,445
350	3490	1,745
377	3835	1,9175

Dovreste studiare attentamente il programma n. 5. Esso fornisce l'esempio più semplice di come si può scrivere un loop di timing.

LOOP DI TIMING

La definizione di *«loop di timing»* è già stata data in questo Capitolo. Questi loop sono largamente impiegati nei programmi dei microprocessori, in sistemi che controllano o mettono in se-

quenza strumenti o macchine. Ecco alcune delle caratteristiche di questi loop:

- Sono richiesti loop di timing con durate di tempo diverse: microsecondi, millisecondi e secondi.
- I loop di timing vengono usati spesso.
- Dato che vengono usati spesso, di solito appaiono nel programma del microcomputer, sotto forma di subroutine.

Prenderemo ora in considerazione una subroutine di questo tipo, una che abbiamo caricato nella PROM partendo dalla posizione di memoria H = 060 e L = 000. Questa subroutine genererà un ritardo di temporizzazione che è precisamente uguale a 0,2 secondi.

Subroutine N. 1

Indirizzo di memoria LO H	*L*	*Byte di istruzione ottale*	*Codice mnemonico*	*Cicli di clock*	*Descrizione*
060	000	021	LXI D	10	Carica i due byte seguenti nei registri E e D, rispettivamente
060	001	301	301		Byte di timing per il registro E
060	002	150	150		Byte di timing per il registro D
060	003	035	DCR E	5	Decrementa di uno il registro E
060	004	302	JNZ	10	Se il registro E è 000_8, ignora questa istruzione; altrimenti, salta all'indirizzo di memoria dato nei due byte seguenti
060	005	003			Byte d'indirizzo LO
060	006	060			Byte d'indirizzo HI
060	007	025	DCR D	5	Decrementa di uno il registro D
060	010	302	JNZ	10	Se il registro D è 000_8, ignora questa istruzione; altrimenti, salta all'indirizzo di memoria dato nei due byte seguenti
060	011	003			Byte d'indirizzo LO
060	012	060			Byte d'indirizzo HI
060	013	015	DCR C	5	Decrementa di uno il registro C
060	014	302	JNZ	10	Se il registro C è 000_8, ignora questa istruzione; altrimenti, salta all'indirizzo di memoria dato nei due byte seguenti
060	015	000			Byte d'indirizzo LO
060	016	060			Byte d'indirizzo HI
060	017	311	RET	10	Rientro incondizionato dalla subroutine

La Fig. 5-5 mostra un diagramma di flusso per questa subroutine di 0,2 secondi. Notate che nella subroutine si entra con un byte di timing già presente nel registro C. Come si può vedere dal diagramma di flusso, vi è un loop del registro E all'interno di un loop del registro D, che è all'interno di un loop del registro C. Segue ora un programma che impiega questa subroutine.

Programma N. 6

Indirizzo di memoria LO	*Byte di istruzione ottale*	*Codice mnemonico*	*Cicli di clock*	*Descrizione*
100	061	LXI D	10	Carica i due byte seguenti nello stack pointer
101	200	200		Byte d'indirizzo LO
102	000	000		Byte d'indirizzo HI
103	016	MVI C	7	Trasferisci il byte seguente nel registro C
104	•	•		Byte di timing per il registro C

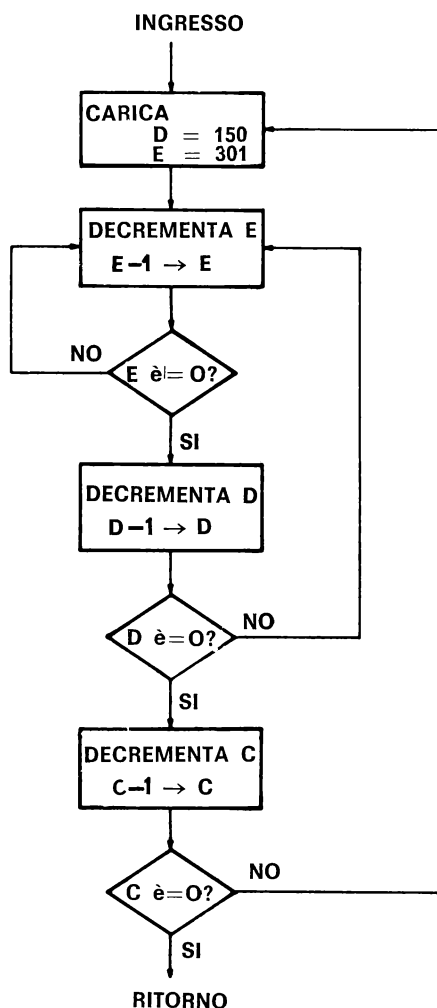



Fig. 5-5. Diagramma di flusso per la subroutine n. 1 che genera un ritardo di 0,200 secondi.

105	323	OUT	10	Genera un impulso di selezione dispositivo che setta il flip-flop 7476
106	000	000		Codice dispositivo
107	315	CALL	17	Chiama in modo incondizionato la subroutine posizionata all'indirizzo di memoria dato dai due byte seguenti
110	000			Byte d'indirizzo LO
111	060			Byte d'indirizzo HI
112	323	OUT	10	Genera un impulso di selezione dispositivo che azzera il flip-flop 7476
113	001	001		Codice dispositivo, diverso
114	166	HLT	7	Alt

Supponiamo che il byte di timing per il registro C all'indirizzo di memoria H = 000 e L = 001 sia *001.* Il passo che si verifica sia nel programma n. 6 che nella subroutine n. 1 si può così riassumere:

a. Lo stack pointer all'interno dell'8080 a HI = 000 e LO = 200. Viene spostato un byte di timing nel registro C. Questo occupa un totale di diciassette cicli di clock. Comunque, il contatore a cinque decadi collegato al flip-flop 7476 non misura questo tempo.
b. Viene generato un impulso di selezione dispositivo tramite un'istruzione OUT. Questa setta il flip-flop 7476 a livello logico 1 e da l'avvio al contatore.
c. Viene chiamata in modo incondizionato una subroutine all'indirizzo di memoria H = 060 e LO = 000. Questo occupa diciassette cicli clock, o 8,5 μs, un tempo che viene misurato dal contatore.
d. Nella subroutine, i byte di temporizzazione vengono spostati nei registri D ed E. Questo occupa dieci cicli di clock.
e. Viene stabilito un loop di timing nel registro E. I contenuti di questo registro vengono decrementati 193 volte. E' richiesto un totale di 193 x 15 = 2895 cicli di clock.
f. I contenuti del registro D sono decrementati di uno. Questo occupa quindici cicli di clock.
g. Viene stabilito un secondo loop di timing con il registro E, solo questa volta i contenuti di questo registro vengono decrementati 256 volte. Questo occupa 3840 cicli di clock.
h. I contenuti del registro D sono decrementati di uno. Questo occupa quindi cicli di clock.
i. Vengono ripetuti i passi *«g»* e *«h»* per 103 volte. I cicli di clock richiesti in totale sono 3840 x 103 + 15 x 103 cicli di clock.
j. I contenuti del registro C sono decrementati di uno. Questo occupa cinque cicli di clock. Dato che il registro C è ora 000_8, il flag di zero viene settato al livello logico 1.

k. Si usano dieci cicli di clock con l'istruzione JNZ alla posizione di memoria H = 060 e L0 = 014.
l. Vi è un rientro incondizionato dalla subroutine. Questo occupa dieci cicli di clock.
m. Infine, viene generato un impulso di selezione dispositivo tramite un'istruzione OUT. Questo occupa dieci cicli di clock. Il flip-flop 6476 viene azzerato a livello logico 0 ed il conteggio si arresta.

Quanti cicli di clock vi sono in tutto? Facciamo il calcolo:

a.		i.	397.065
b.		j.	5
c.	17	k.	10
d.	10	l.	10
e.	2.895	m.	10
f.	15		

Il numero totale di cicli di clock è 400.037. A 500 ns per ogni ciclo di clock, questo si traduce in un ritardo di 0,2000185 secondi. La subroutine di timing è programmata in maniera tale che, se il byte di timing per il registro C fosse **002**, il numero totale di cicli di clock sarebbe 800,037, o 0,4000185 al secondo. Ogni volta che il registro C viene settato ad un bit di più della volta precedente, viene aggiunto 0,2 secondi addizionale al ritardo di timing. Vedrete che una subroutine che genera un ritardo di 0,2 secondi è molto comune. Il massimo ritardo che potete generare è di 51,2000185 secondi, usando solo i registri C, D ed E. Diamo un esempio di ritardi di temporizzazione tipici:

Byte di timing per il registro C	*Tempo di delay (secondi)*
001	0,2000185
002	0,4000185
005	1,0000180
012	2,0000185
031	5,0000185
062	10,0000185
144	20,0000185
372	50,0000185

Mentre il massimo ritardo di tempo è qualcosa in più di 50 secondi, possono essere necessari ritardi più lunghi. Questi ritardi di tempo maggiori si possono ottenere nidificando le subroutine in modo che una subroutine di ritardo di 30 secondi viene usata 120 volte per fornire un ritardo complessivo di un'ora. Questo procedimento viene attuato per mezzo del programma n. 7.

Programma N. 7

Indirizzo di memoria LO	*Byte di istruzione ottale*	*Codice mnemonico*	*Cicli di clock*	*Descrizione*
000	061	LXI SP	10	Carica i due byte seguenti nello stack pointer
001	200	200		Byte d'indirizzo LO
002	000	000		Byte d'indirizzo HI
003	001	LXI B	10	Carica i due byte seguenti nei registri C e B, rispettivamente
004	226	226		Byte del registro C
005	170	170		Byte del registro B
006	315	CALL	17	Chiama in modo incondizionato la subroutine posizionata all'indirizzo di memoria dato dai due byte seguenti.
007	000			Byte d'indirizzo LO
010	060			Byte d'indirizzo HI
011	005	DCR B	5	Decrementa di uno il registro B
012	302	JNZ	10	Se il registro B è 000_8, ignora questa istruzione; altrimenti, saltare all'indirizzo di memoria dato nei due byte seguenti
013	006			Byte d'indirizzo LO
014	000			Byte d'indirizzo HI
015	166	HLT	7	Alt

Qui usiamo la subroutine n. 1 per generare un ritardo di tempo di 30 secondi. Il byte di timing C ha un valore di 226_8, che corrisponde al decimale 150. Questo ritardo di temporizzazione di 301 s viene chiamato 120 volte, o 170_8 in codice ottale, per produrre un ritardo totale di un'ora.

Questo viene fatto con solo 30 byte di istruzioni posizionati in memoria. Per ottenere il risultato desiderato, usiamo il principio della *nidificazione* dei loop:

Nesting (Nidificazione) — In un computer, l'inserimento di un loop o di una routine all'interno di un loop o di una routine più grossi.

COME SI ATTUANO LE SEQUENZE IN UN MICROCOMPUTER

Nel Capitolo 4 e in questo Capitolo, avete imparato come:

- Generare impulsi di selezione dispositivo individuali, fino ad un totale di 512 diversi impulsi.
- Scrivere routine a ritardo di tempo che creano dei ritardi varianti da 5 μs a minuti o perfino ad ore.
- Cambiare lo stato di un flip-flop J-K esterno per attivare e disattivare un dispositivo digitale, come un contatore.

- Scrivere loop di programma che ripeteranno una sequenza di operazioni dopo un intervallo di tempo predeterminato.

Così, siete già in grado di creare una varietà di sequenziatori programmabili, nei quali mettete in sequenza una serie di operazioni a intervalli di tempo presettati che sono determinati da ritardi di tempo programmati. Ecco alcune definizioni:

Sequenzial operation (Operazione sequenziale)	L'eseguire delle operazioni una dopo l'altra.
Sequencer (Sequenziatore)	Un dispositivo elettronico che può essere settato per dare inizio ad una serie di eventi e per far seguire gli eventi in sequenza, cioè in ordine.
Programmable sequencer (Sequenziatore programmabile)	Un sequenziatore nel quale l'ordine degli eventi e delle operazioni può essere cambiato con l'aiuto della programmazione.

Questi concetti vengono dettagliatamente trattati nel Capitolo 5 del *Bugbook I.* Nella Fig. 5-6, vedete un esempio di come gli impulsi di selezione dispositivo e le routine di tempo possono essere usati per eseguire operazioni sequenziali. In questo esempio, vengono versati in un grosso serbatoio due liquidi, la soluzione risultante viene riscaldata e mescolata allo scopo di dar luogo ad una reazione, infine i contenuti del serbatoio vengono filtrati. Sono presenti otto dispositivi diversi; un interruttore per l'alimentazione, tre valvole, tre motori e un riscaldatore. Vengono usati due impulsi di selezione dispositivo per ogni dispositivo: uno per attivarlo e l'altro per disattivarlo. Sono necessari molti diversi ritardi di tempo, da 0,01 ora, o 36 secondi, a 65,4 minuti e possono essere tutti facilmente programmati nel microcomputer. Il ritardo di 36 secondi viene usato almeno sette volte. Il punto importante in questo sistema è costituito dal fatto che *la sequenza di operazioni e gli intervalli di tempo fra una e l'altra si possono cambiare facilmente alterando semplicemente il programma del microcomputer.* Questo è un altro esempio di software che sostituisce l'hardware. Per mettere in sequenza le operazioni, non sono necessari contatori o circuiti logici speciali. Il microcomputer può fare il tutto con l'aiuto di impulsi di selezione dispositivo, di otto flip-flop, di otto buffer TTL, e di otto relè optoelettronici che controllano la corrente alternata con segnali a livello logico 0 e 1 da un circuito TTL. Dovrebbe essere chiaro il fatto che queste idee vengono estese a semafori, alla protezione degli edifici, ad operazioni di macchine ed a molte altre situazioni che richiedono operazioni sequenziali.

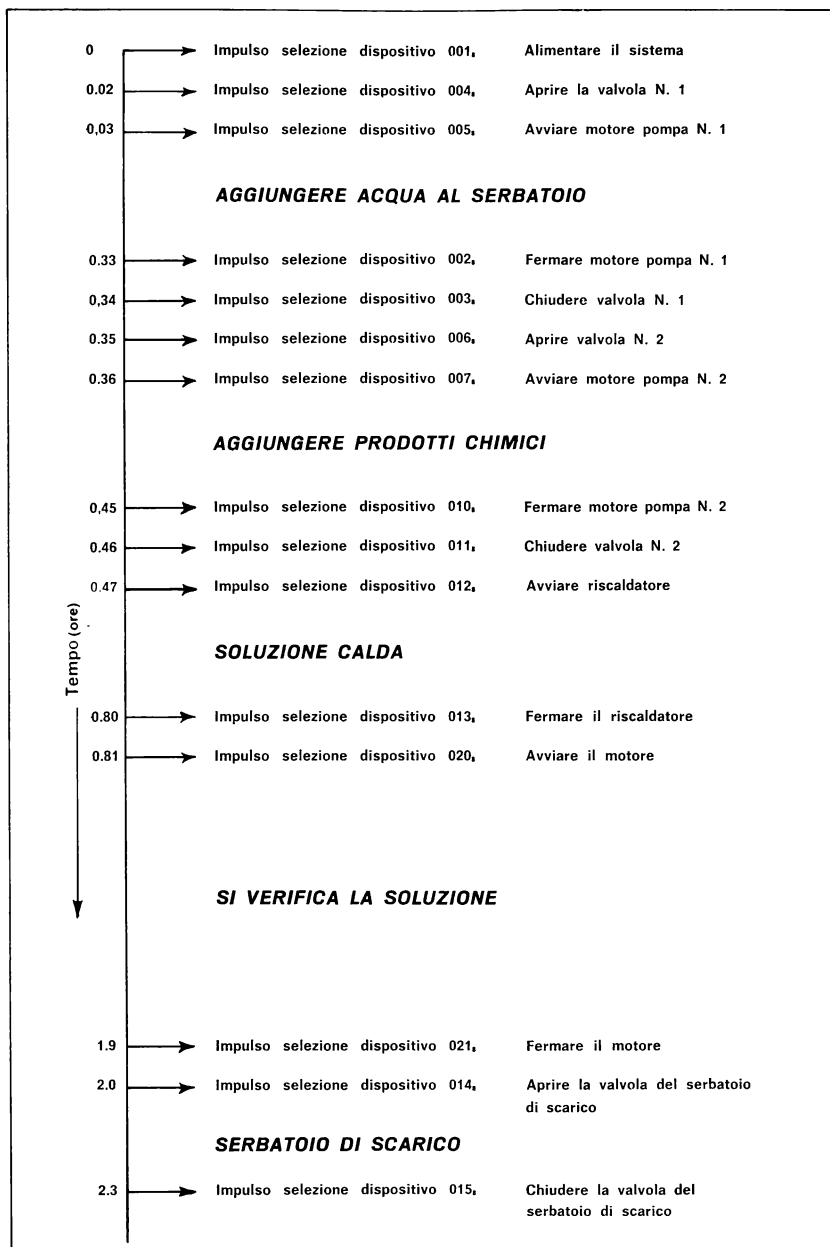



Fig. 5-6. Una sequenza di operazioni associate con un reattore batch, parte di un'attrezzatura usata dai chimici per eseguire reazioni chimiche.

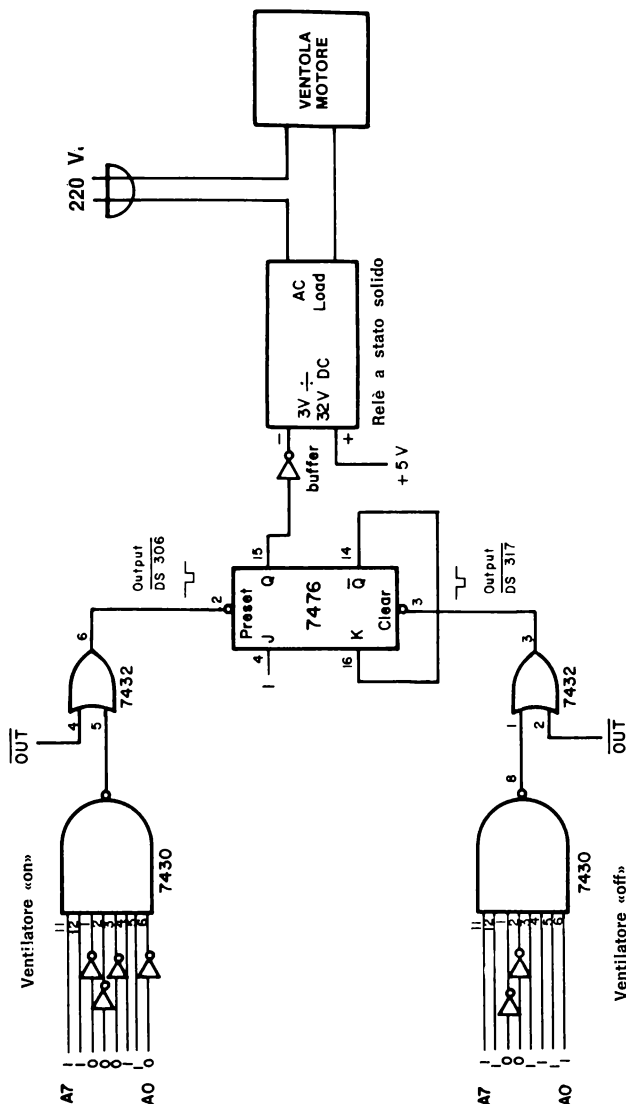



Fig. 5-7. Questo circuito mostra (a) l'uso di porte NAND 7430 ad 8 ingressi, come decodificatori dei codici dispositivo ad 8 bit generati dal microcomputer e (b) l'uso di un relè a stato solido, isolato otticamente, come driver di un motore di un ventilatore, con un segnale in ingresso TTL.

COME SI CONTROLLA L'ALIMENTAZIONE DI UN MICROCOMPUTER

La Fig. 5-7 mostra un circuito nel quale un flip-flop 7476 controlla la corrente alternata al motore di un ventilatore. Ciò av-

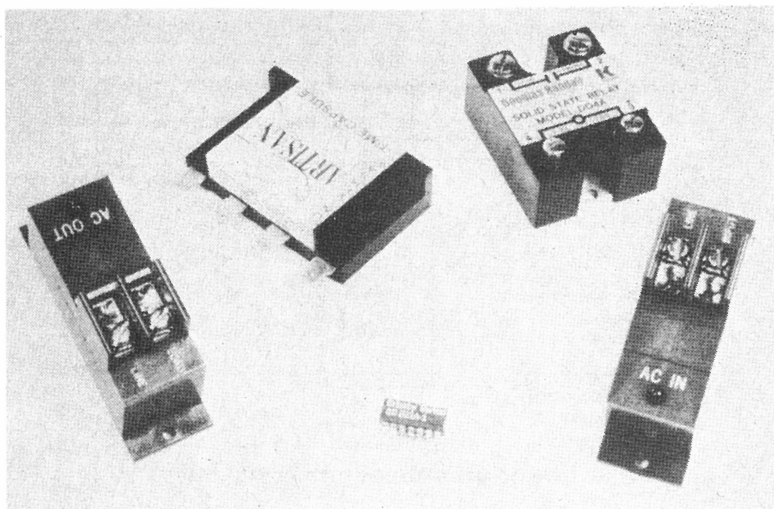



Fig. 5-8. Tipici relè a stato solido e ad isolamento che sono disponibili ad un prezzo di \$ 10-\$ 20 l'uno. Tali relè possono controllare fino a 10 A a 115 V da una sola uscita TTL. Il relè a destra in basso converte la corrente alternata in un segnale di corrente continua, permettendovi così di rivelare quando un motore di un ventilatore o un dispositivo che consuma corrente è attivo. Il chip piccolo non è un semiconduttore, ma un reed relè, che può operare da + 5 V TTL. L'«Aristan time capsule» è un dispositivo di delay a stato solido.

viene con l'aiuto di un relè a stato solido e ad isolamento ottico (Fig. 5-8), un tipo di relè che è diventato molto comune negli ultimi anni. Un semplice ingresso TTL di questo relè può facilmente controllare 10 A alla corrente alternata di 115 V. Anche se non parleremo in questo libro di tali relè, dovrebbe essere chiaro che un microcomputer può facilmente controllare il ventilatore. L'indirizzo di memoria LO è collegato ad ognuno dei due gate NAND 7430 ad otto ingressi, che decodificano il codice dispositivo a 8 bit prodotto dall'istruzione OUT. All'unico stato di livello logico 0 prodotto dal gate 7430 viene effettuata un'operazione di OR con l'impulso $\overline{\text{OUT}}$.

TEST

Questo test prova quanto avete capito circa i concetti e gli esperimenti di questo Capitolo. Per favore scrivete le risposte su di un foglio di carta a parte.

5-1. Dare la definizione dei seguenti termini: loop, loop di timing, periodo, ciclo di clock e stato.

5-2. Disegnare un semplice circuito digitale che può conteggiare dieci cicli di clock.

5-3. Scrivere un programma che genererà ritardi di timing che sono multipli di circa 100 μs.

La vostra prova sarà accettabile se sarete in grado di rispondere a tutte le domande suddette in modo corretto in un'ora di tempo, a libro chiuso. Incontreremo spesso loop di timing nei programmi del microcomputer.

CHE COSA AVETE REALIZZATO IN QUESTO CAPITOLO?

All'inizio di questo capitolo era stato stabilito che, alla fine, avreste dovuto essere in grado di:

- Dare la definizione di: loop, loop di timing, periodo, ciclo di clock e stato.
  Questo è stato fatto in vari punti del capitolo.
- Programmare un loop di timing che sia in grado di generare ritardi di temporizzazione che sono multipli di 0,2 secondi.
  Lo avete fatto nella subroutine n. 1.
- Programmare un loop di timing che sia in grado di generare ritardi di timing che sono multipli di circa 0,5 ms.
  Lo avete fatto nel programma n. 6.
- Dimostrare in che modo un microcomputer può agire come multivibratore monostabile.
  Avete generato impulsi di timing singoli con l'aiuto del programma del microcomputer e del flip-flop 7476.

CAPITOLO 6

Generazione delle Informazioni di Stato

Si va delineando sempre più il fatto che una delle funzioni importanti del microprocessore 8080 è quella di controllare il trasferimento di otto bit di informazioni digitali, chiamati byte, fra il bus di dati interno, situato entro il chip stesso dell'8080, e il bus di dati esterno, posto all'esterno del chip. Internamente, le informazioni possono essere memorizzate in registri quali l'accumulatore, i registri universali, il contatore di programma, lo stack pointer, e il registro istruzioni. Esternamente, le informazioni vengono memorizzate nella memoria, nei latch di stato, e nei dispositivi di ingresso/uscita. In questo capitolo, impareremo qualcosa di più su questo argomento. Arriverete a farvi un'idea complessiva del modo in cui il microprocessore opera e interagisce con il «mondo esterno» al chip.

Questo è un capitolo alquanto avanzato, e che potete saltare per il momento se il vostro tempo è limitato Se volete, potete leggere prima i Capitoli 7 e 8, in quanto riveleranno la vostra abilità nell'interfacciare. Potrete poi ritornare a questo capitolo; pur saltandolo, non vi mancherà nessuna informazione necessaria.

OBIETTIVI

Alla fine di questo capitolo, sarete in grado di:

- Spiegare le differenze fra bus di dati interno ed esterno in un sistema microcomputer basato sull'8080.
- Fare un elenco delle sorgenti e delle destinazioni delle informazioni che appaiono sul bus di dati esterno.

- Spiegare le differenze fra uno stato, un ciclo di clock, un ciclo di istruzioni e un ciclo macchina.
- Spiegare che cosa è un bit di stato ed un byte di stato.
- Descrivere i nove diversi tipi di cicli macchina.
- Descrivere la funzione di ognuno degli otto diversi bit di stato.
- Spiegare in che modo si possono combinare in modo logico le uscite di controllo sul chip dell'8080 con uno o più bit di stato, e fornire almeno un esempio di una combinazione logica di questo tipo.
- Disegnare diagrammi di timing che rivelano il comportamento delle istruzioni tipiche del microcomputer. Tali diagrammi dovrebbero chiaramente dimostrare gli stati logici degli ingressi di controllo, delle uscite di controllo e dei bit di stato più importanti.
- Spiegare i diversi tipi di dati che possono apparire sul bus di dati esterno.
- Spiegare i timing per i cicli macchina tipici.
- Spiegare come si attua la tecnica del passo-passo in un microcomputer 8080, completando il tutto con diagrammi di timing e schemi di circuiti.
- Discutere le caratteristiche della porta di ingresso/uscita a otto bit 8212.

DEFINIZIONI

Bus — Un canale su cui vengono trasferite le informazioni digitali, da una qualunque delle molte sorgenti ad una qualunque delle molte destinazioni. Può avvenire solo un trasferimento di informazioni alla volta. Mentre avviene tale trasferimento, tutte le altre sorgenti che sono collegate al bus devono essere disabilitate.

To bus (Verbo) — Collegare fra di loro molti dispositivi digitali che ricevono o trasmettono segnali digitali per mezzo di un set di canali di conduzione, chiamati bus, sul quale vengono trasferite tutte le informazioni fra uno e l'altro di questi dispositivi.

Control input (Ingresso di controllo) — Un pin di ingresso sul chip dell'8080 che controlla il comportamento del microprocessore.

Control output (Uscita di controllo) — Un pin di uscita sul chip dell'8080 che controlla il comportamento dei chip e dei dispositivi esterni collegati al chip dell'8080.

Data bus buffer/latch (Buffer/latch del bus di dati) — Un latch a 8 bit con uscite three-state, che controllano il trasferimento delle informazioni fra il bus di dati interno al microprocessore 8080 e il bus di dati esterno.

Execution (Esecuzione)	Una delle due parti funzionali di un ciclo istruzione.
External bus (Bus esterno)	Il bus di dati bidirezionale a 8 bit esterno al microprocessore 8080 e al quale sono collegati la memoria, i latch e i dispositivi di uscita e i buffer d'ingresso dei dispositivi d'ingresso.
Fetch	Una delle due parti funzionali di un ciclo istruzione. L'insieme delle azioni di acquisizione di un indirizzo di memoria, di un'istruzione o di un byte di dati dalla memoria.
Instruction cycle (Ciclo istruzione)	Un gruppo successivo di cicli macchina, da uno a cinque, che eseguono insieme una sola istruzione all'interno del microprocessore.
Internal bus (Bus interno)	Un bus di dati bidirezionale situato all'interno del microprocessore 8080 ed al quale sono collegati l'accumulatore, il registro istruzioni, il registro universale, un registro temporaneo e l'unità logico/aritmetica.
Machine cycle (Ciclo macchina)	Una suddivisione di un ciclo istruzione durante il quale avviene un gruppo di azioni correlate, all'interno del microprocessore. Nel microprocessore 8080, esistono nove diversi cicli macchina. Tutte le istruzioni sono combinazioni di uno o più di questi cicli macchina.
Nonoverlapping two-phase clock (Clock a due fasi senza sovrapposizione)	Un clock a due fasi in cui gli impulsi di clock delle fasi individuali non si sovrappongono.
State (Stato)	Un intervallo costante uguale in lunghezza al periodo del clock che guida l'unità centrale.
Status bit (Bit di stato)	Un solo bit di informazione in uscita che viene posto sul bus di dati esterno durante l'esecuzione di un ciclo macchina e su cui viene effettuato un latch da parte di un circuito integrato chiamato latch di stato. Dato che questo bit viene acquisito presto dal latch, può essere usato per controllare gli eventi esterni che si verificano più tardi nel ciclo macchina.
Status byte (Byte di stato)	Un byte a 8 bit, cioè un'unità di 8 bit, che contiene otto diversi bit di stato.
Status latch (Latch di stato)	Un circuito integrato, ad esempio l'Intel 8212, che effettua un latch sugli otto bit di stato quando essi appaiono sul bus di dati esterno.
Three-state device	Un dispositivo logico a semiconduttore, in cui esistono tre possibili stati di uscita: (1) uno sta-

(Dispositivo a tre stati) to di «livello logico 0», (2) uno stato di «livello logico 1», o (3) uno stato in cui l'uscita è, in effetti, scollegata dal resto del circuito e non ha alcuna influenza su di esso.

Two-phase clock (Clock a due fasi) Un dispositivo di timing a due uscite che fornisce due serie continue di impulsi di timing che sono sincronizzati insieme, con un solo impulso di clock dalla seconda serie che segue sempre un solo impulso di clock dalla prima serie. A seconda del tipo di clock a due fasi, gli impulsi della prima e della seconda serie possono o meno sovrapporsi l'un l'altro.

IL BUS DATI BIDIREZIONALE

Il *bus dati bidirezionale a 8 bit* è il principale collegamento in trasmissione dati fra l'accumulatore del microprocessore 8080 e la memoria, i dispositivi d'ingresso, i dispositivi di uscita, e il *latch di stato.* Per bidirezionale si intende che i dati possono fluire in entrambe le direzioni sul bus, dal chip ad un dispositivo e da un dispositivo nel chip. Il bus è *three-state.* Nella Fig. 6-1 vedete un diagramma del flusso dei dati.

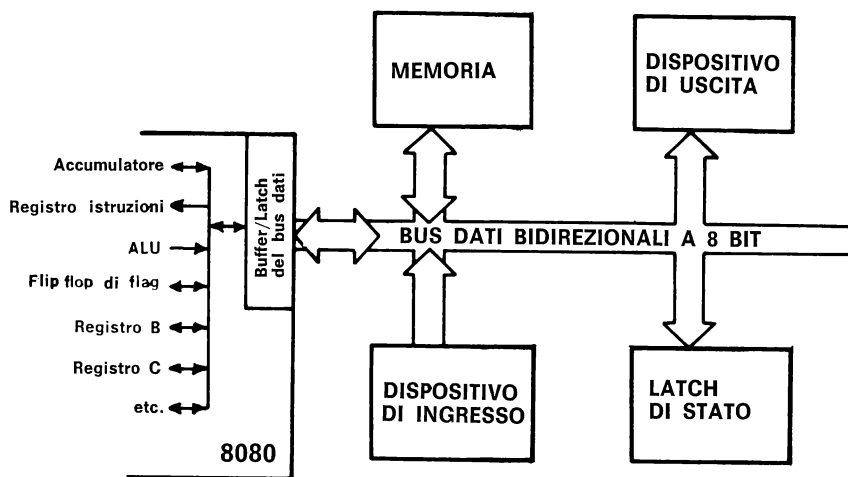



Fig. 6-1. Il buffer/latch del bus di dati all'interno dell'8080 funziona da interfaccia interna fra il bus di dati interno e il bus di dati bidirezionale esterno.

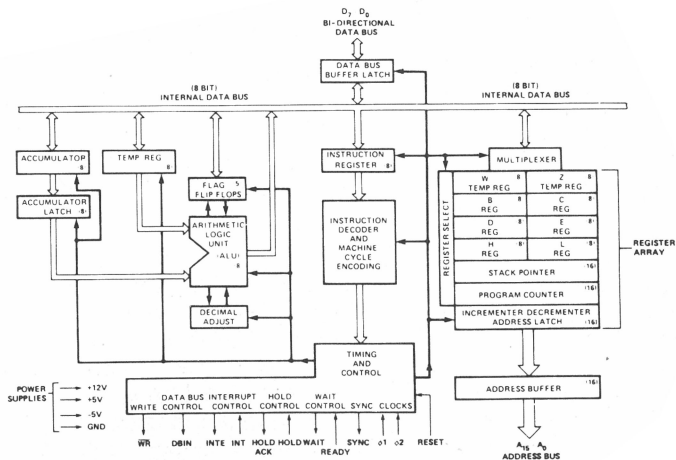
Parlando del *bus* nel Capitolo 7 del *Bugbook II,* sono state date queste definizioni:

Bus Un canale sul quale vengono trasferite le informazioni digitali, da una qualunque delle molte

sorgenti ad una qualunque delle molte destinazioni. Può avvenire solo un trasferimento di informazioni alla volta. Mentre avviene questo trasferimento, tutte le altre sorgenti che sono collegate al bus devono essere disabilitate.

To bus (Verbo) — Collegare fra loro molti dispositivi digitali che ricevono o trasmettono segnali digitali per mezzo di un comune set di canali di conduzione, chiamati bus, sul quale vengono trasferite tutte le informazioni fra uno e l'altro di tali dispositivi.

Three-state device (Dispositivo a tre stati) — Un dispositivo logico a semiconduttore in cui vi sono tre possibili stati di uscita: (1) uno stato di «livello logico 0», (2) uno stato di «livello logico 1», o (3) uno stato in cui l'uscita è, in effetti, scollegata dal resto del circuito e non ha alcuna influenza su di esso.

Nella definizione di «bus» che è stata data, il particolare importante è che *può avvenire solo un trasferimento di informazioni alla volta.* Dato che un bus è un gruppo di canali condiviso da più entità, nascerebbe il caos se tutti coloro che trasmettono sul bus tentassero di inviare le informazioni simultaneamente. Lo scopo principale di un bus è quello di minimizzare il numero di interconnessioni richieste per trasferire le informazioni fra i dispositivi digitali. Minore è il numero di interconnessioni, più facile sarà eseguire il layout delle piastre a circuito stampato. Anche all'interno del circuito integrato stesso, si usano *bus interni* per facilitare la fabbricazione dei chip. Il microprocessore 8080 ha un bus interno e comunica con i dispositivi esterni su di un *bus esterno* presente sui circuiti stampati.

Courtesy Intel Corp.

Nella Fig. 6-1 vedete uno schema a blocchi funzionale del bus di dati bidirezionale esterno a 8 bit. Segue ora uno schema a blocchi funzionale del chip stesso dell'8080. Notate il *buffer/latch del bus di dati* nella parte centrale in alto dello schema. Questa parte del chip 8080 è il buffer fra il bus di dati interno e quello esterno. Quando il latch viene abilitato, i dati possono passare in entrambe le direzioni fra i due bus di dati.

Sul bus di dati bidirezionale a 8 bit, (il bus esterno), vi sono tre trasmettitori:

- memoria
- qualunque dispositivo d'ingresso, e
- il buffer/latch del bus di dati all'interno del chip dell'8080;

e quattro ricevitori:

- memoria
- qualunque dispositivo di uscita,
- il latch di stato e
- il buffer/latch del bus di dati interno al chip dell'8080.

Le informazioni che appaiono sul bus di dati esterno possono essere:

- Un byte di dati che viene trasferito dalla memoria, attraverso il buffer/latch del bus di dati, nell'accumulatore (o in uno dei sei registri universali);
- Un byte istruzioni che viene trasferito dalla memoria, attraverso il buffer/latch del bus di dati, nel registro istruzioni;
- Un byte di dati che viene trasferito da un dispositivo d'ingresso, attraverso il buffer/latch del bus di dati, nell'accumulatore;
- Un byte di dati proveniente dall'accumulatore che viene trasferito, attraverso il buffer/latch del bus di dati, verso un dispositivo di uscita;
- Un byte di stato che viene trasferito, attraverso il buffer/latch del bus di dati, nel latch di stato; e
- Un byte istruzioni proveniente da un dispositivo d'ingresso, che viene trasferito, durante una condizione d'interruzione, attraverso il buffer/latch del bus di dati, nel registro istruzioni;

e forse altre ancora. Chiaramente, il microprocessore ha dei notevoli problemi di calcolo dei tempi: *esso deve coordinare il trasferimento delle informazioni fra tutti i trasmettitori e tutti i ricevitori sul bus di dati bidirezionale esterno a 8 bit.* Esso ha lo stesso identico problema sul bus di dati interno. Come risolve tutto ciò il microprocessore? Questo sarà l'argomento di questo capitolo. Questa discussione vi darà una «sensazione» di quello che succede all'interno del microcomputer senza farvi scendere nei molti sottili dettagli delle operazioni del microprocessore. Per

ulteriori particolari, vedere la letteratura della Intel Corporation, e più precisamente l'*8080 Microcomputer System Manual* e l'*Intellec 8/Mod 80 Microcomputer Development System Reference Manual,* che forniscono entrambi informazioni utili e diagrammi di timing su quanto avviene durante i diversi *cicli macchina* del microcomputer.[8, 9] Una parte del materiale che segue è tratto da questi due manuali, per gentile concessione della Intel Corporation.

CICLI ISTRUZIONE

Citiamo l'*Intellect 8/Mod 80 Microcomputer Development System Reference Manual:*

L'8080 è comandato da un clock a due fasi, ad una frequenza massima di 2,08 MHz. Tutte le attività di elaborazione si riferiscono al periodo di questo clock. Le due fasi di clock che non si sovrappongono, indicate con $\varnothing_1$ e $\varnothing_2$, vengono fornite dall'insieme dei circuiti esterni. Il clock $\varnothing_1$ divide il ciclo di elaborazione in *«stati».* Uno stato è la più piccola unità dell'attività di elaborazione (480 ns quando il processore opera alla massima velocità) e viene definita come l'intervallo fra due transizioni successive che vanno in senso positivo del clock $\varnothing_1$. La logica di timing all'interno dell'8080 usa gli ingressi del clock per produrre un impulso SYNC, che identifica il primo stato di ogni ciclo macchina...

«... Come mostra la Fig. 6-2, il segnale SYNC è correlato al fronte principale del clock $\varnothing_2$. Vi è un ritardo fra la transizione da basso ad alto di $\varnothing_2$ e il fronte positivo dell'impulso SYNC. Vi è anche un ritardo corrispondente fra l'impulso $\varnothing_2$ successivo e il fronte di caduta del segnale SYNC. Le informazioni di stato sono visualizzate da D0 a D7 durante questo stesso intervallo. La commutazione dei segnali di stato (avviene solo quando $\varnothing_2$ è a livello logico 1) ...».

«Un *ciclo istruzione* consiste di due parti funzionali, il *fetch* e l'*esecuzione.* Ognuna di queste parti funzionali, a turno, consiste di un numero di *cicli macchina.* Durante il fetch, un'istruzione selezionata (uno, due o tre byte) viene estratta dalla memoria e depositata nel registro istruzioni della CPU. Durante l'esecuzione, l'istruzione è decodificata e tradotta in attività di elaborazione specifiche. La routine di fetch richiede un ciclo macchina per ogni byte che deve essere prelevato dalla memoria. La durata della parte di esecuzione riguardante il ciclo istruzione *dipende dal tipo di istruzione che è stata prelevata dalla memoria.* Alcune istruzioni non richiedono nessun altro ciclo macchina, a parte quelli necessari per prelevare l'istruzione dalla memoria; altre istruzioni, comunque, richiedono cicli macchina addizionali per scrivere o leggere i dati nella o dalla memoria o dai dispositivi di I/O. L'istruzione DAD è un'eccezione per il fatto che richiede due cicli macchina addizionali per completare la somma di una coppia interna di registri.

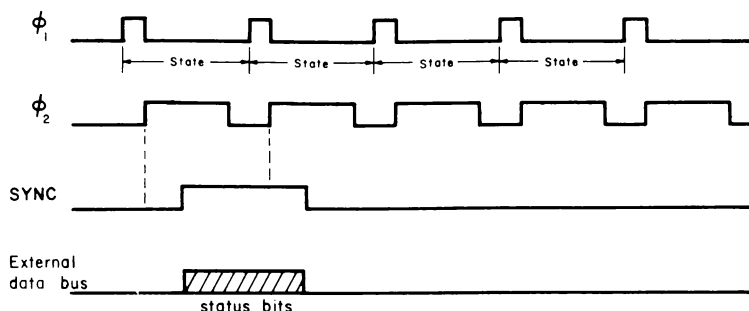



Fig. 6-2. Timing dei clock $\varnothing_1$ $\varnothing_2$ e di SYNC. SYNC non si verifica nel secondo e terzo ciclo macchina della istruzione DAD.

«Ogni ciclo istruzione contiene uno, due, tre, quattro o cinque cicli macchina. Ogni ciclo macchina, a sua volta, consiste di tre, quattro, o cinque stati. Uno stato viene definito come un intervallo costante, uguale in lunghezza al periodo dell'oscillatore di clock che comanda la CPU (una fase). Cioè, uno stato viene sempre definito così, tranne che in tre casi. Le eccezioni alla regola sono lo stato di WAIT, lo stato di hold (HLDA), e lo stato di alt (HLTA); ... Dopo una breve considerazione, ne capirete la ragione, dato che gli stati di WAIT, HLDA e HLTA dipendono da eventi esterni e sono, per loro natura, di lunghezza indeterminata. Osservate, comunque, che anche questi stati eccezionali devono essere sincronizzati con gli impulsi del clock che comanda. Quindi le durate di tutti gli stati, compresi questi, sono multipli interi della fase di clock.

«Riassumendo, ogni *fase* di clock segna uno *stato;* da tre a cinque stati costituiscono un *ciclo macchina;* e da uno a cinque cicli macchina formano un *ciclo istruzione.* Un intero ciclo istruzioni richiede, in qualunque punto, da quattro a diciotto fasi per essere completato (da 2 a 9 microsecondi), a seconda del tipo di istruzione coinvolta».

Quanto detto finora, è scritto bene e non ha bisono di essere molto elaborato. Il punto importante è che un microprocessore è un dispositivo elettronico digitale fornito di clock. Gli impulsi di clock sono necessari per sincronizzare e fare in modo che si verifichino specifiche operazioni. All'interno del microprocessore, deve avvenire un numero sufficiente di operazioni per quelle istruzioni che non è possibile eseguire tutte con un solo treno di clock. Quindi, è necessario un clock a due fasi con impulsi $\varnothing_1$ e $\varnothing_2$ che non si sovrappongono. Per i cicli di istruzioni più semplici, *sono disponibili otto impulsi di clock individuali per coordinare le azioni all'interno del microprocessore.* Per le istruzioni semplici, questo è tutto quello che serve. Comunque, per le istruzioni più complicate, *sono richiesti impulsi di clock addizionali per coordinare le azioni del microprocessore.* L'istruzione più complicata nel set

di istruzioni dell'8080 è l'istruzione XTHL, che richiede diciotto stati, o 36 impulsi di clock individuali, per essere completata. Trattate il microprocessore 8080 come *un circuito digitale che potreste, volendo, montare su breadboarding.* Per voi, questo potrebbe non essere comodo, a causa dei molti collegamenti che sarebbero necessari. Ciononostante, non dovreste considerare la struttura interna del chip dell'8080 come un mistero. Quello che c'è all'interno è semplicemente un circuito digitale munito di clock che viene attivato per mezzo di una coppia di treni di clock sincronizzati, cioè clock a due fasi. Molti dei circuiti interni al chip dell'8080 sono bus oriented, per cui servono meno collegamenti di fili di quanti non vi aspettereste normalmente.

La Fig. 6-3 presenta un gruppo di tre istruzioni del microcomputer, SUB A, OUT <B2>, e CALL <B2> <B3>, che richiedono rispettivamente quattro, dieci e 17 stati (o cicli di clock). Di questa figura parleremo più dettagliatamente nel paragrafo successivo.

CICLI MACCHINA

Nella Fig. 6-3 vi è un gruppo di tre istruzioni, ognuna delle quali richiede il suo proprio ciclo di istruzione. L'istruzione SUB A ha un ciclo macchina ed un totale di quattro stati. L'istruzione OUT <B2>, a due byte, richiede tre cicli macchina ed un totale di dieci stati. Infine, l'istruzione a tre byte CALL <B2> <B3> richiede cinque cicli macchina ed un totale di diciassette stati. I diversi cicli machina all'interno di ogni istruzione sono etichettati M_1, M_2, ... M_5. Cinque cicli macchina è il numero massimo richiesto per una qualunque delle tre istruzioni mostrate nella figura. In questo paragrafo, parleremo dei cicli macchina individuali all'interno di tutto il ciclo istruzioni.

Vi sono nove tipi di cicli macchina che possono verificarsi all'interno di un ciclo istruzioni, sebbene in qualunque ciclo istruzione dato, non appariranno più di cinque cicli macchina. Questi tipi di cicli macchina sono:

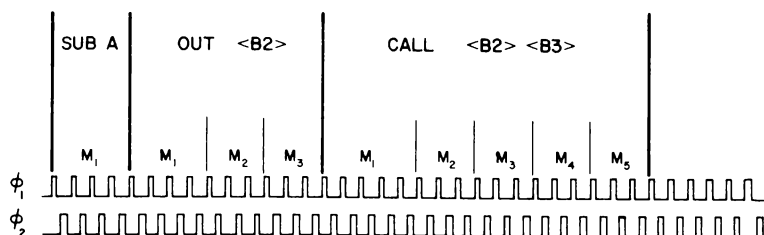



Fig. 6-3. Stato, ciclo macchina e timing del ciclo istruzione. Sono mostrate tre istruzioni: SUB A, OUT <B2>, e CALL <B2> <B3>, che sono, rispettivamente, istruzioni ad un solo byte, a due byte, a tre byte e che richiedono quattro, dieci e diciassette cicli di clock.

- *Fetch* — Questo ciclo macchina consiste di quattro o cinque stati, ad eccezione degli stati di WAIT, HLDA, HLTA, che contengono qualunque numero intero di stati maggiore di tre. Durante questo ciclo, viene acquisito l'indirizzo di memoria, vengono messi a disposizione i bit di stato sul bus di dati esterno a 8 bit, avviene il trasferimento tra registri all'interno dell'8080, e vengono eseguite semplici operazioni logiche o aritmetiche. Per alcune istruzioni, questo è l'unico ciclo macchina richiesto. Qualunque istruzione che richieda solo quattro o cinque stati, richiede solo il ciclo macchina di fetch. Questo ciclo macchina prende il suo nome dal fatto che, durante il ciclo, il codice operativo per l'istruzione viene «fetched» (prelevato) dalla posizione di memoria presente nel contatore di programma. Questo codice operativo viene trasferito nel registro istruzione all'interno del chip dell'8080, dove viene in seguito decodificato dal decodificatore di istruzione in una serie di azioni che vengono eseguite dal microcomputer. Durante questo ciclo, il contatore di programma viene incrementato di uno, dando così la posizione del byte di istruzione seguente.
- *Memory read (Lettura in memoria)* — Questo ciclo macchina consiste di tre stati. Durante questo ciclo, un byte presente nella locazione di memoria indicata dal contatore di programma viene trasferito dalla memoria ad uno dei registri all'interno del chip dell'8080. Tali registri comprendono l'accumulatore, B, C, D, E, H, L.
- *Memory write (Scrittura in memoria)* — Questo ciclo macchina consiste di tre o quattro stati, durante i quali i contenuti di un registro vengono trasferiti nella locazione di memoria rilevata dai registri H ed L. I registri pointer H e L possono anche essere incrementati o decrementati durante questo ciclo macchina.
- *Output (Uscita)* — Questo ciclo macchina consiste di tre stati, durante i quali viene messo a disposizione il codice dispositivo d'ingresso sul bus dell'indirizzo di memoria a 16 bit, nonché i contenuti dell'accumulatore sul bus di dati bidirezionale esterno a 8 bit.
- *Input (Ingresso)* — Anche questo ciclo macchina consiste di tre stati, durante i quali viene messo a disposizione il codice dispositivo d'ingresso sul bus dell'indirizzo di memoria a 16 bit, e il buffer/latch del bus di dati all'interno del microprocessore 8080 viene abilitato per permettere ai dati in ingresso di apparire sul bus esterno, per essere trasferiti nell'accumulatore.
- *Stack write (Scrittura nello stack)* — Durante questo ciclo macchina a tre stati, viene posto un byte di dati a 8 bit sul bus di dati esterno e trasferito alla locazione di memoria

data o da M (SP-1) o da M (SP-2), dove SP è il registro stack pointer interno al microprocessore 8080.

- *Stack read (Lettura nello stack)* — Durante questo ciclo macchina a tre stati, viene trasferito un byte di dati a 8 bit dalla locazione di memoria data o da M (SP) o da M (SP + 1), tramite il bus di dati esterno, ai registri del microprocessore, quali H, L, B, C, D, E o il contatore di programma.
- *Halt (Alt)* — Questo ciclo macchina può contenere un qualunque numero di stati intero maggiore di tre. Il microprocessore rimane in uno stato di WAIT finché l'ingresso READY del chip dell'8080 è a livello logico 0. L'uscita WAIT del chip dell'8080 è a livello logico 1 per permettere di riconoscere che la CPU è in uno stato di WAIT.
- *Interrupt (Interruzione)* — Questo ciclo macchina a cinque stati assomiglia al ciclo macchina di fetch, ad eccezione del fatto che il contatore di programma, che è già stato incrementato nel ciclo di fetch, non viene incrementato. Questo fa in modo che il valore precedente alla interruzione del contatore venga conservato nello stack e permette un rientro al programma interrotto dopo che la richiesta d'interruzione è stata elaborata.

I seguenti cicli macchina si verificano per le istruzioni presenti nella Fig. 6-3.

Fetch: M_1 in SUB A, OUT <B2>, e CALL <B2> <B3>
Lettura in memoria: M_2 in OUT <B2> e sia M_2 che M_3 in CALL <B2> <B3>
Uscita: M_3 in OUT <B2>
Scrittura nello stack: sia M_4 che M_5 in CALL <B2> <B3>

Potreste chiedervi perché si richiedono due cicli macchina simili durante lo stesso ciclo istruzione, come avviene nell'istruzione CALL. La risposta è che le informazioni vengono trasferite da o verso registri diversi interni al chip dell'8080 durante i cicli macchina di lettura in memoria e di scrittura nello stack; di conseguenza, sono necessari in ogni caso due cicli diversi. Non è difficile immaginare che i microprocessori del futuro, che saranno tutti molto più veloci dell'8080, richiederanno più di cinque cicli macchina per eseguire istruzioni ancora più complicate, come una moltiplicazione o una divisione.

Ulteriori informazioni sui cicli macchina si possono trovare nei Riferimenti 8 e 9. Entrambe queste fonti sono servite per la stesura di questo paragrafo.

IDENTIFICAZIONE DEI CICLI MACCHINA

Una domanda che molte persone si pongono quando iniziano a studiare il microprocessore 8080 è: come si può determinare qua-

le ciclo macchina comprende un'istruzione? La letteratura della Intel descrive cicli macchina tipici per istruzioni quali INP, OUT, ADD, CALL, HALT, e altre, ma queste descrizioni sono lontane dall'essere un inventario completo di tutto il set di 78 istruzioni. Possiamo credere di poter decifrare la maggior parte delle istruzioni avendo tempo e pazienza sufficienti, ma esiste un modo di farlo mentre il microcomputer sta operando?

La risposta alla domanda precedente è che è realmente possibile determinare quale ciclo macchina comprende un'istruzione; lo si può fare mentre il microcomputer opera. I dettagli verranno forniti nei paragrafi che seguono. La maggior parte delle informazioni sono contenute nei Riferimenti 8 e 9.

Come si Effettua un Latch sui Bit di Stato

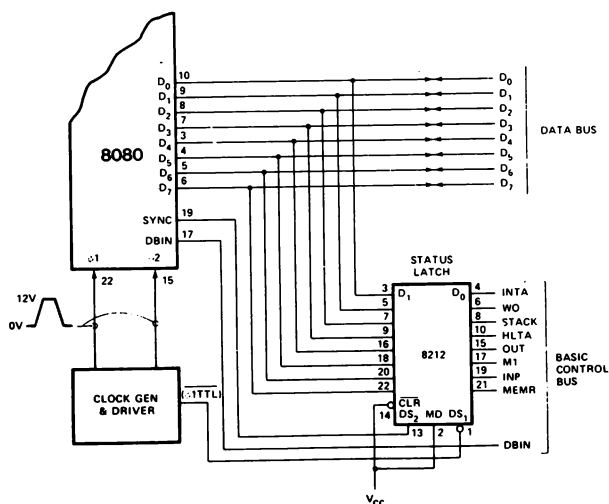
Come fa il microcomputer a identificare il ciclo macchina in corso? *Lo fa trasmettendo un segnale di stato a 8 bit durante il primo stato, T_1, di ogni ciclo macchina.* Questi segnali di stato appaiono sul bus di dati bidirezionale esterno a 8 bit, e viene effettuato su di essi un latch dal *circuito del latch di stato* 8212 mostrato nella Fig. 6-4. Vi sono otto bit di stato, e ne parleremo brevemente. L'importante è sapere che *i bit di stato sono le prime informazioni poste sul bus di dati esterno durante un nuovo ciclo istruzione.* Perciò, i bit di stato precedono un'istruzione o un byte di dati provenienti dalla memoria o un byte di dati in ingresso o in uscita sul bus di dati esterno. Tali informazioni vengono dopo i bit di stato, non prima.

Courtesy Intel Corp.

Fig. 6-4. Schema del circuito utilizzato per effettuare un latch sugli otto bit di stato che appaiono sul bus di dati esterno durante gli stati T_1 e T_2.

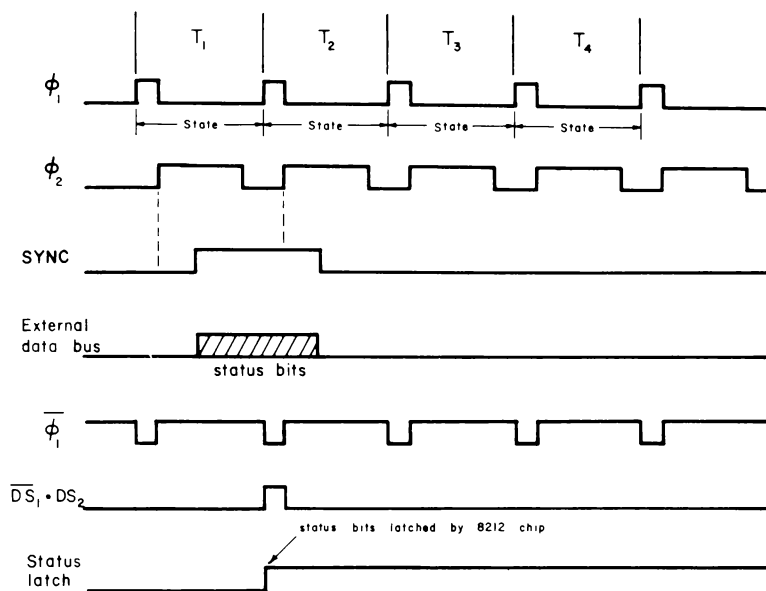



Fig. 6-5. Diagramma di timing che illustra in che modo viene effettuato un latch sui bit di stato presenti sul bus di dati esterno a 8 bit, da parte del chip 8212.

Come si effettua un latch sui bit di stato? Ve lo mostrano sia la Fig. 6-4 che la Fig. 6-5. Osservate nella Fig. 6-4 che l'uscita SYNC dell'8080 è collegata all'ingresso DS_2 del chip 8212. Quando DS_2 è a livello logico 1 e $\overline{DS}_1$ è a livello logico 0, viene effettuato un latch sulle informazioni di stato disponibili sul bus di dati esterno, ed esse appaiono ai pin di uscita 4, 6, 8, 10, 15, 17, 19 e 21 dell'8212. L'ingresso al pin 1 sul chip 8212 è $\varnothing_1$; quando questo ingresso va a livello logico 0, con SYNC a livello logico 1, viene effettuato il latch (Fig. 6-5). L'impulso $\overline{DS}_1$ e DS_2 ha un'ampiezza d'impulso di 500 ns. Un tipico microcomputer 8080 ha un latch di questo tipo, come quello mostrato nella Fig. 6-6.

Gli Otto Bit di Stato

Gli otto bit di stato che appaiono sul bus di dati esterno durante gli stati T_1 e T_2 di un ciclo macchina hanno i seguenti simboli e definizioni:

Simbolo	*Bit del bus di dati*	*Descrizione*
INTA	D0	Segnale di riconoscimento della richiesta di INTERRUPT, al verificarsi del quale esso è a livello logico 1. Questo segnale dovrebbe essere usato per effettuare un gate su di un'istruzione di ripristino, RST, sul bus di dati quando DBIN (pin 17 sul chip 8080) è a livello logico 1

$\overline{WO}$	D1	Se l'operazione nel ciclo macchina in corso è una funzione di USCITA o di SCRITTURA di memoria, questo stato sarà a livello logico 0. Se l'operazione è una LETTURA di memoria o un INGRESSO, questo stato è a livello logico 1
STACK	D2	Un livello logico 1 indica che il bus degli indirizzi di memoria a 16 bit contiene l'indirizzo dello stack dallo stack pointer, SP
HLTA	D3	Segnale di riconoscimento di un'istruzione di HALT. Quando il ciclo macchina è un'istruzione di HALT, questo bit di stato sarà a livello logico 1
OUT	D4	Un livello logico 1 per questo bit di stato indica che il bus degli indirizzi di memoria a 16 bit contiene il codice dispositivo a 8 bit del dispositivo di uscita e che il bus di dati esterno conterrà i dati in uscita quando WR (pin 18 sul chip 8080) è a livello logico 0
M_1	D5	Questo bit di stato fornisce un segnale di livello logico 1 per indicare che la CPU è nel ciclo di fetch per il primo byte di un'istruzione
INP	D6	Un livello logico 1 per questo bit di stato indica che il bus dell'indirizzo di memoria a 16 bit contiene il codice dispositivo a 8 bit per il dispositivo d'ingresso e che i dati in ingresso dovrebbero essere posti sul bus di dati esterno quando DBIN (pin 17 sul chip 8080) è a livello logico 1
MEMR	D7	Un livello logico 1 indica che il bus di dati esterno riceverà i dati provenienti dalla memoria durante questo ciclo macchina

Citiamo il Riferimento 8:

«I cicli macchina che si verificano in un particolare ciclo istru-

Courtesy Tychon, Inc.

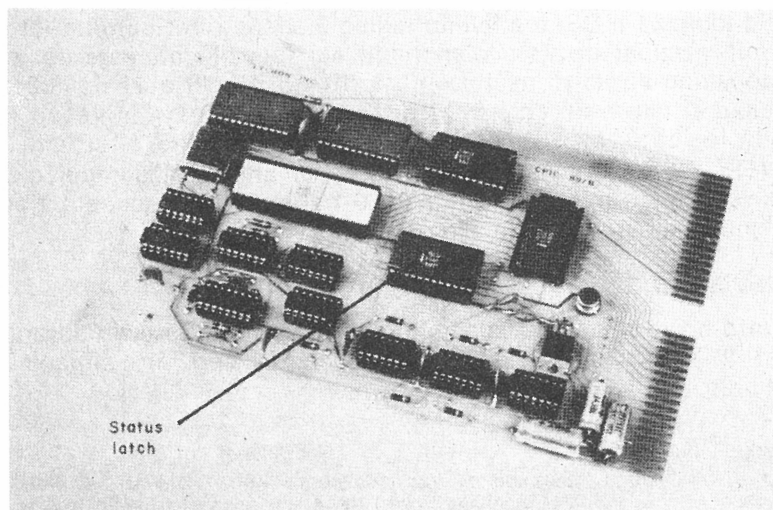



Fig. 6-6. La posizione del latch di stato a 8 bit, sulla piastra a circuito stampato CPIC-80/B. Questa piastra è parte di un microcomputer 8080 costruito dalla E & L Instruments, Inc.

zione dipendono dal tipo di istruzione, fermo restando il fatto che il primo ciclo macchina in qualunque ciclo istruzione è sempre un FETCH.

«Il processore identifica il ciclo macchina in corso trasmettendo un segnale di stato a 8 bit durante il primo stato di ogni ciclo macchina. Le informazioni di stato aggiornate appaiono sulle linee dell'8080 (da D0 a D7) durante l'intervallo SYNC. Questi dati possono essere conservati in latch, decodificati, e usati per sviluppare segnali di controllo per circuiti esterni. La Tabella 6-1 mostra in che modo vengono distribuite le informazioni di stato sul bus di dati del microprocessore.

«*I segnali di stato vengono forniti principalmente per il controllo dei circuiti esterni.* La semplicità delle interfacce, piuttosto che l'identificazione delle macchine, costituiscono la definizione logica dei bit di stato individuali. Osserverete quindi che determinati cicli macchina del processore vengono identificati unicamente da un solo bit di stato, ma che altri non lo sono. Il bit di stato di M_1, D5, ad esempio, identifica senza ambiguità un ciclo macchina di FETCH. Uno STACK READ, invece, è indicato dalla coincidenza dei segnali di STACK e di MEMR. I dati di identificazione dei cicli macchina possono essere validi anche nel test e nelle fasi di messa a punto dello sviluppo del sistema».

Come indicato nella precedente citazione, per identificare un ciclo macchina specifico, usiamo la tabella della verità mostrata nella Tabella 6-1.

Ingressi e Uscite di Controllo dell'8080

Nel Capitolo 1, abbiamo parlato degli ingressi e delle uscite di controllo da e verso il microprocessore 8080. Gli ingressi di controllo sono: (Fig. 6-7):

RESET (pin 12)
INT (pin 14)
READY (pin 23)
HOLD (pin 13)

e quelli di uscita sono:

WAIT (pin 24)
$\overline{\text{WR}}$ (pin 18)
HLDA (pin 21)
INTE (pin 16)
SYNC (pin 19)
DBIN (pin 17)

Gli ingressi e le uscite di controllo suddetti possono essere combinati in modo logico con le otto uscite di stato:

INTA
$\overline{WO}$
STACK
HLTA
OUT
M_1
INP
MEMR

Tabella 6-1. Tabella della Verità Relativa al Tipo di Ciclo Macchina nei Confronti dei Bit di Stato Individuali nella Parola di Stato a 8 Bit che Appare sul Bus di Dati Durante lo Stato T_1

Tipo di Ciclo Macchina	Bit del Bus Dati ed Informazioni di Stato							
	MEMR D7	INP D6	M_1 D5	OUT D4	HLTA D3	STACK D2	$\overline{WO}$ D1	INTA D0
Instruction Fetch	1	0	1	0	0	0	1	0
Memory Read	1	0	0	0	0	0	1	0
Memory Write	0	0	0	0	0	0	0	0
Stack Read	1	0	0	0	0	1	1	0
Stack Write	0	0	0	0	0	1	0	0
Input	0	1	0	0	0	0	1	0
Output	0	0	0	1	0	0	0	0
Interrupt	0	0	1	0	0	0	1	1
Halt	1	0	0	0	1	0	1	0
Interrupt While Halt	0	0	1	0	1	0	1	1

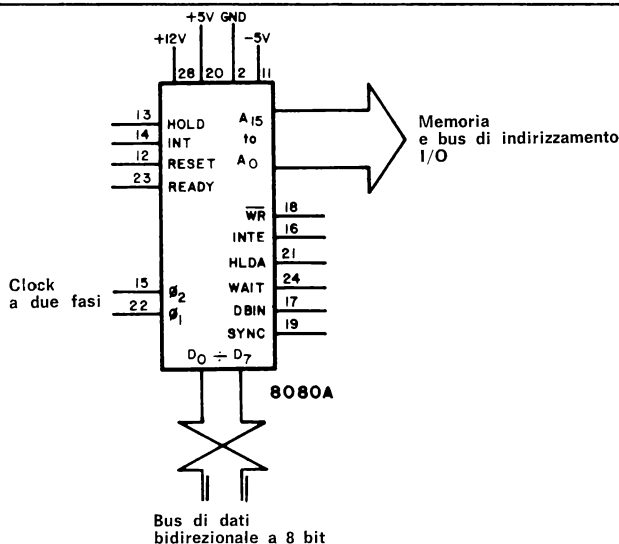



Fig. 6-7. Schema a blocchi del microprocessore 8080 che mostra gli ingressi e le uscite di controllo, nonché il bus degli indirizzi di memoria e il bus di dati bidirezionale a 8 bit (il bus di dati esterno).

su cui viene effettuato un latch dal buffer/latch 8212 per coordinare e controllare il trasferimento dei dati da e verso il microprocessore. Questo è un punto molto importante. Non dovete limitarvi ai quattro pin degli ingressi di controllo e a sei pin delle uscite di controllo sul chip dell'8080 (Fig. 6-7); si possono usare anche i bit di stato su cui è stato effettuato il latch. Le uscite di strobe $\overline{IN}$ e $\overline{OUT}$ non sono posizionate sul chip dell'8080; li ricavate dai bit di stato dopo che su di essi è stato effettuato il latch. L'attuazione dei latch sui bit di stato è una parte utile e importante della

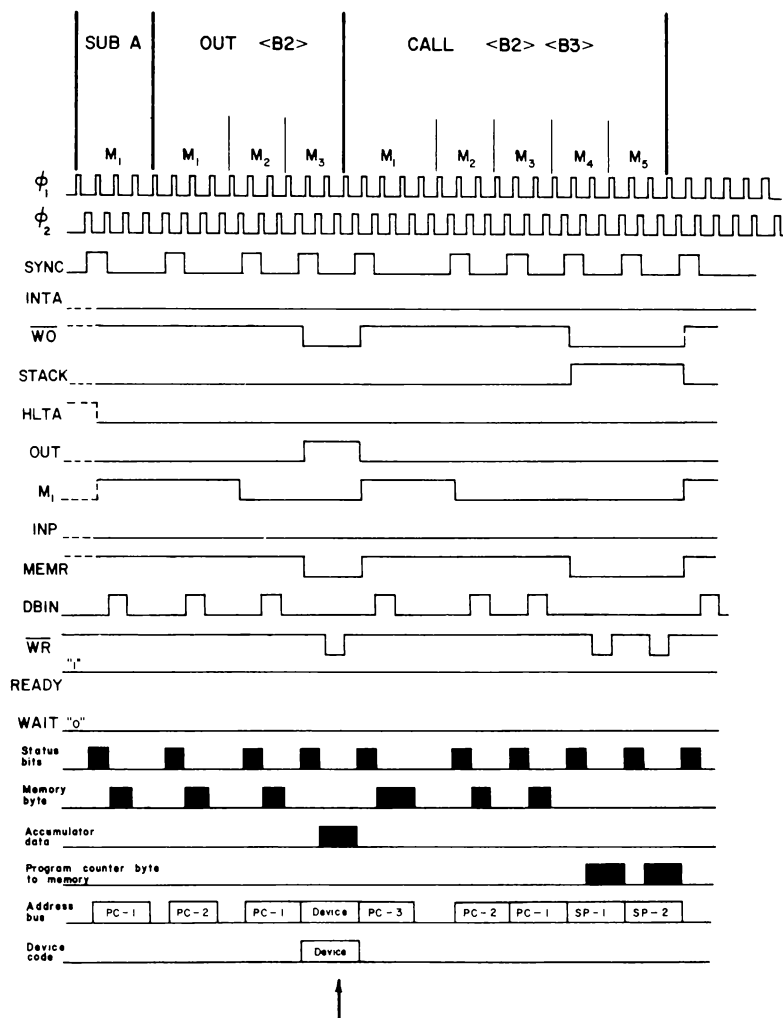



Fig. 6-8. Diagrammi di timing per i segnali presenti sul bus di dati, sul bus degli indirizzi e sul bus di controllo. Notate i cambiamenti nelle uscite del bit di stato.

progettazione in elettronica digitale, che sfrutta al massimo l'uso del bus di dati ed elimina il bisogno di pin di uscita in più sul microprocessore. Anche se può sembrare che il prezzo che pagate per questa capacità è uno stato per ogni ciclo macchina per effettuare un latch sui bit di stato, ricordatevi che durante lo stesso stato, l'indirizzo di memoria a 16 bit viene caricato nel contatore di programma.

Diagrammi di Timing per le Istruzioni Tipiche dell'8080

Parleremo ora dei punti summenzionati con l'aiuto della Fig. 6-8, che mostra i diagrammi di timing per i seguenti segnali digitali:

- Il clock a due fasi dell'8080 e parecchi ingressi e uscite di controllo (Fig. 6-9).

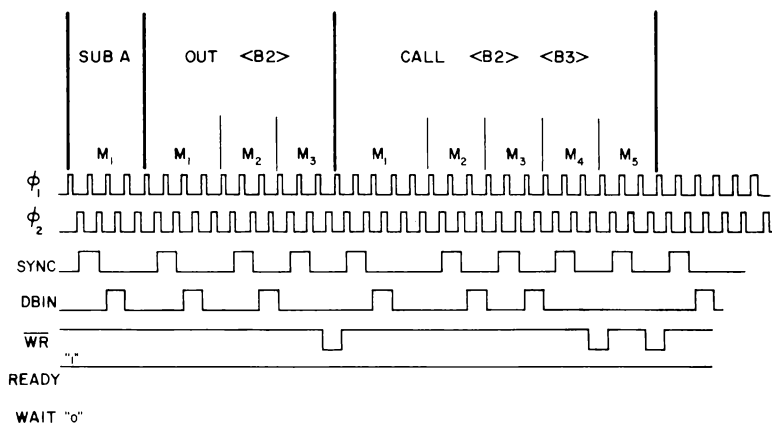



Fig. 6-9. Gli ingressi del clock a due fasi e molti dei segnali d'ingresso e di uscita che si trovano sul microprocessore 8080.

- I bit di stato su cui viene effettuato un latch (Fig. 6-10).
- I diversi tipi di dati che appaiono sul bus di dati esterno (Fig. 6-11).

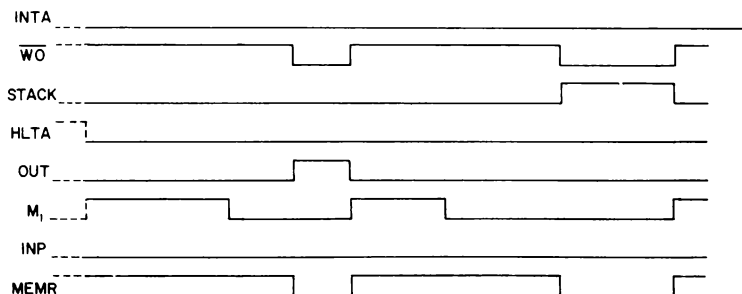



Fig. 6-10. Gli otto bit di stato su cui viene effettuato un latch.

- Le informazioni relative agli indirizzi di memoria ed al codice dispositivo di I/O che appaiono sul bus degli indirizzi di memoria (Fig. 6-12).

Le lettere PC e SP stanno per contatore di programma e stack pointer, rispettivamente; sono registri a 16 bit interni al microprocessore 8080. Per «PC-1» si intende che l'informazione sul bus degli indirizzi esterno a 16 bit, ha un valore di uno meno del contenuto del contatore di programma. *Il contatore di programma contiene l'indirizzo dell'istruzione da eseguire successivamente, non l'indirizzo dell'istruzione che viene eseguita in quel momento.*

Le tre istruzioni di cui parleremo sono: SUB A, un'istruzione ad un solo byte che azzera l'accumulatore; OUT <B2>, un'istruzione a due byte che fornisce un impulso di strobe $\overline{\text{OUT}}$ che permette ad un dispositivo esterno di effettuare un latch sui contenuti dell'accumulatore, che appare per un breve istante sul bus di dati esterno; e CALL <B2> <B3>, un'istruzione a tre byte che effettua un salto incondizionato ad una subroutine all'indirizzo di memoria dato dai byte B2 e B3. Segue ora un semplice programma che utilizza queste istruzioni uno dopo l'altra:

Indirizzo di memoria LO	*Istruzione ottale*	*Descrizione*
000	227	Sottrai i contenuti dell'accumulatore, cioè azzera l'accumulatore
001	323	Genera un impulso di selezione dispositivo per mettere in uscita otto bit di dati dell'accumulatore sul dispositivo con il codice dispositivo dato nel byte seguente
002	000	Codice dispositivo per il dispositivo di uscita
003	315	Richiamo incondizionato della subroutine situata alla locazione di memoria data nei due byte seguenti
004	040	Byte d'indirizzo di memoria LO
005	000	Byte d'indirizzo di memoria HI
006	166	Alt
040	311	Ritorno incondizionato da questa subroutine al programma principale

Andrebbe notato che lo stack pointer, SP, è stato precedentemente posizionato a LO = 200_8 e HI = 000_8 e il programma precedente può essere ripetutamente eseguito semplicemente resettando il

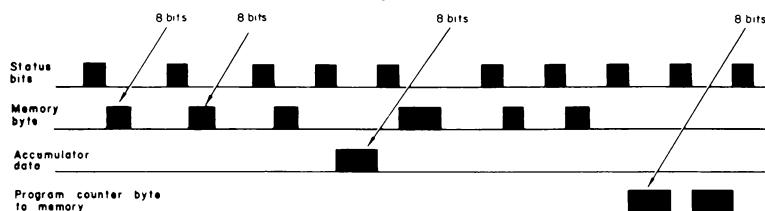



Fig. 6-11. Rappresentazione dei tipi di dati che appaiono sul bus di dati bidirezionale esterno. Osservate che sul bus di dati non sono presenti simultaneamente due tipi diversi di dati.

contatore di programma a HI = 000_8 e LO = 000_8, con l'aiuto dell'interruttore di controllo manuale RESET sul pannello frontale di un microcomputer 8080.

L'esecuzione di questo programma si può riassumere in questo modo:

Bus di indirizzo di memoria LO	*Bus dati Bidirezionale*	*Codice mnemonico*	*MEMR*	*INP*	*M_1*	*OUT*	*HLTA*	*STACK*	*$\overline{WO}$*	*INTA*
			1	0	0	0	1	0	1	0

(*Bit di Stato*: MEMR … INTA)

NOTA: Al momento, il microcomputer è nello stato di HALT. Negli stati che vedete, viene effettuato un latch sugli otto bit di stato. Per far partire il programma, usiamo l'interruttore di controllo RESET per resettare il contatore di programma a HI = 000_8 e LO = 000_8. La ragione per cui vi abbiamo fornito queste informazioni vi apparirà chiara in seguito.

000	227	SUB A	1	0	1	0	0	0	1	0
001	323	OUT	1	0	1	0	0	0	1	0
002	000	<B2>	1	0	1	0	0	0	1	0
000*	000	Uscita dati sul bus	0	0	0	1	0	0	0	0
003	315	CALL	1	0	1	0	0	0	1	0
004	200	<B2>	1	0	1	0	0	0	1	0
005	000	<B3>	1	0	1	0	0	0	1	0
177	000	Scrittura nello stack	0	0	0	0	0	1	0	0
176	006	Scrittura nello stack	0	0	0	0	0	1	0	0
040	311	RET	1	0	1	0	0	0	1	0
177	006	Lettura nello stack	1	0	0	0	0	1	1	0
176	000	Lettura nello stack	1	0	0	0	0	1	1	0
006	166	HLT	1	0	1	0	0	0	1	0
377	377	—	1	0	0	0	1	0	1	0

Dopo l'esecuzione del programma suddetto, il microcomputer è nello stato di HALT. L'istruzione contrassegnata da asterisco (*) mostra il ciclo macchina relativo all'uscita dei dati. Il codice dispositivo per il dispositivo di uscita selezionato a LO = 002_8 è 000_8.

La Fig. 6-8 fornisce i diagrammi di timing per i primi nove cicli macchina dell'elenco del programma suddetto. L'intero programma richiede tredici cicli macchina; nello stato di HALT alla fine del programma non avviene nessun'altra esecuzione. Se studiate bene la figura, arriverete a capire la maggior parte dei concetti importanti delle operazioni di un microprocessore. Vogliamo mettere in rilievo i seguenti punti:

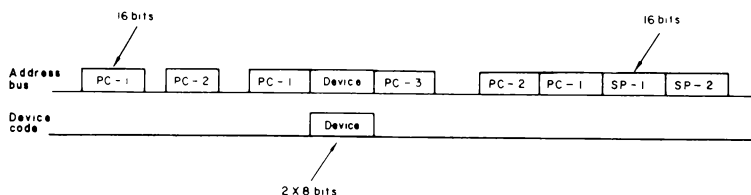



Fig. 6-12. Rappresentazione dei tipi di dati che appaiono sul bus degli indirizzi a 16 bit.

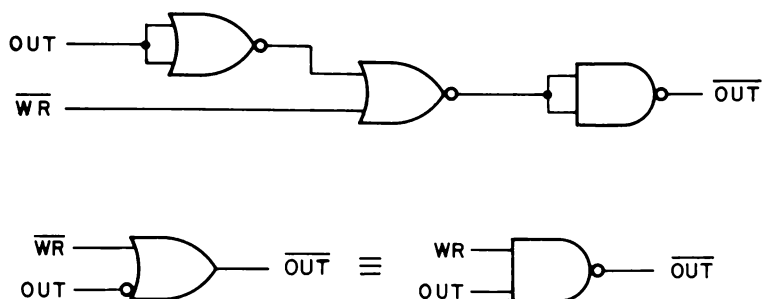



Fig. 6-13. Circuiteria logica per la generazione del segnale di controllo $\overline{OUT}$

- Il momento che precede il primo ciclo macchina, M_1, nell'istruzione SUB A.

  Supponiamo qui che il microcomputer sia nello stato di HALT, in cui sugli otto bit di stato viene effettuato un latch nel chip 8212 negli stati seguenti: INTA = STACK = OUT = M_1 = INP = livello logico 0 e WO = HLTA = MEMR = livello logico 1.

- L'unico momento in cui sul bit di stato OUT viene effettuato un latch da parte del chip 8212 ad un livello logico 1.

  Notate che in nessun altro momento il bit di stato OUT si trova a livello logico 1. Questa condizione vi permette unicamente di determinare due importanti condizioni che si verificano simultaneamente: (1) il codice dispositivo di I/O è presente sotto forma di due byte a 8 bit identici sul bus degli indirizzi di memoria a 16 bit, e (2) i contenuti dell'accumulatore sono presenti sul bus di dati bidirezionale esterno a 8 bit. Queste condizioni si verificano *solo quando sul bit di stato OUT viene effettuato un latch a livello logico 1.* Chiaramente, il bit di stato OUT si può usare per generare gli impulsi di selezione dispositivo che possono fornire impulsi di strobe dall'accumulatore ad uno specifico dispositivo di uscita. Nella Fig. 6-8 è disegnata una freccia verticale verso il fondo della figura per indicare il momento in cui sui dati in uscita dall'accumulatore può essere effettuato un latch da parte di un dispositivo esterno. Un microcomputer 8080 genererà un impulso $\overline{OUT}$ di 500 ns nel periodo in cui il bit di stato OUT è a livello logico 1. Nella Fig. 6-13 vedete l'insieme di circuiti logici usati per questo, coè un semplice gate NAND a due ingressi che viene abilitato solo quando WR e OUT sono entrambi a livello logico 1. L'ampiezza d'impulso di 500 ns è una conseguenza dell'ingresso $\overline{WR}$ che rimane a livello logico 0 solo per 500 ns, come si può vedere nella Fig. 6-7.
  Anche se l'istruzione d'ingresso non si vede, essa funziona in modo simile a quella di uscita. Viene effettuato un latch sul bit di stato INP da parte dell'8212, e viene generato un impulso di selezione dispositivo, mentre viene effettuato un latch su di esso con l'aiuto del circuito logico mostrato nella Fig. 6-14.
  L'impulso di clock negativo, $\overline{IN}$, viene generato per 500 ns quando sia DBIN che INP sono a livello logico 1.

- Il bit di stato M_1 è a livello logico 1 solo durante il primo ciclo macchina di un'istruzione.
- Il bit di stato $\overline{WO}$ è a livello logico 0 quando i dati vengono scritti in memoria o in un dispositivo di uscita. Questo bit di

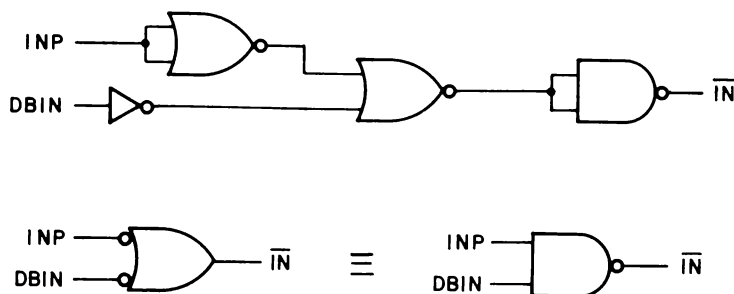



Fig. 6-14. Circuiteria logica per la generazione del segnale di controllo $\overline{\text{IN}}$.

stato è a livello logico 1 mentre viene letto un byte istruzioni dalla memoria. Durante un'istruzione a più byte, si continua ad effettuare un latch su $\overline{\text{WO}}$, a livello logico 1, finché tutta l'istruzione a più byte è stata letta dalla memoria.

- L'uscita SYNC corrispondente al pin 19 del chip 8080 è a livello logico 1 nello stesso momento in cui gli otto bit di stato sono presenti sul bus di dati esterno.
- Sul bit di stato HLTA viene effettuato un latch a livello logico 1. solo quando il microprocessore è nello stato di alt.
- L'uscita $\overline{\text{WR}}$ corrispondente al pin 18 del chip 8080 è a livello logico 0 solo quando i dati vengono scritti in memoria o in un dispositivo di uscita. Quando $\overline{\text{WR}}$ è a livello logico 0, il bit di stato MEMR deve essere a livello logico 0 (dato che non vengono letti dati dalla memoria).
- Un livello logico 1 al bit di stato MEMR indica che il bus di dati esterno riceverà i dati provenienti dalla memoria durante questo ciclo macchina.
- L'uscita DBIN corrispondente al pin 17 sul chip 8080, è a livello logico 1 solo quando i dati provenienti dalla memoria o da un dispositivo d'ingresso sono presenti sul bus di dati esterno. Questo è il segnale che indica che i dati stanno arrivando dalla memoria nell'8080.
- Sul bit di stato STACK viene solitamente effettuato un latch a livello logico 0. Questo avviene a livello logico 1 solo quando il bus degli indirizzi a 16 bit contiene l'indirizzo di una locazione di memoria sullo stack.
- Sul bus di dati esterno possono apparire almeno quattro tipi di byte a 8 bit: gli otto bit di stato, un byte dati o istruzione ad 8 bit proveniente dalla memoria, otto bit di dati provenienti dall'accumulatore, o un byte d'indirizzo a 8 bit che viene trasferito dalla memoria al contatore di programma. *Sul bus di dati esterno appare solo un byte a 8 bit per volta.* Non ci sono

circostanze in cui due diversi byte a 8 bit siano presenti simultaneamente sul bus di dati esterno.

- Il contatore di programma contiene sempre l'indirizzo di memoria a 16 bit del byte istruzione da eseguire successivamente. L'indirizzo di memoria del byte istruzione esistente è sempre di valore di uno meno rispetto a quello che il contatore di programma contiene in quel momento.
- Il codice dispositivo appare sul bus degli indirizzi di memoria sotto forma di due byte a 8 bit identici, uno che occupa il byte d'indirizzo LO e l'altro che occupa il byte d'indirizzo HI.
- Il numero di istruzioni eseguite in un programma si può conteggiare semplicemente applicando l'uscita DBIN al pin 17 sul chip 8080 e il bit di stato M_1, su cui è stato effettuato un latch, ad un gate AND a due ingressi 7408. Ogni impulso dell'uscita del gate corrisponde ad una sola istruzione.

Lo schema della Fig. 6-15 mostra un latch di stato collegato ad un microprocessore 8080. Alcuni segnali di stato non vengono usati spesso. Fra i segnali più importanti derivati dal latch di stato e dai segnali di controllo dell'8080 ci sono $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{INTA}}$, $\overline{\text{I/O R}}$, $\overline{\text{I/O W}}$. In questo libro, abbiamo usato la lettera $\overline{\text{IN}}$ per $\overline{\text{I/O R}}$ e $\overline{\text{OUT}}$ per $\overline{\text{I/O W}}$. I segnali di controllo $\overline{\text{MEMR}}$ e $\overline{\text{MEMW}}$ indicano un'operazione di lettura o di scrittura in memoria.

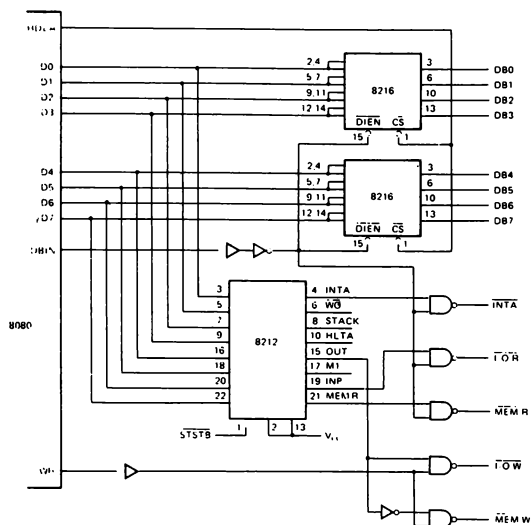
I segnali di controllo $\overline{\text{IN}}$, $\overline{\text{OUT}}$, $\overline{\text{INTA}}$ sono già stati menzionati.

Courtesy Intel Corp.

Fig. 6-15. Schema di un tipico circuito di controllo per l'8080, che usa un chip 8212. $\overline{\text{I/O R}}$ nello schema equivale a $\overline{\text{IN}}$, e $\overline{\text{I/O W}}$ equivale a $\overline{\text{OUT}}$.

I chip 8216 e 8212 della Fig. 6-15 sono stati eliminati in seguito all'introduzione, da parte della Intel Corporation, del system con-

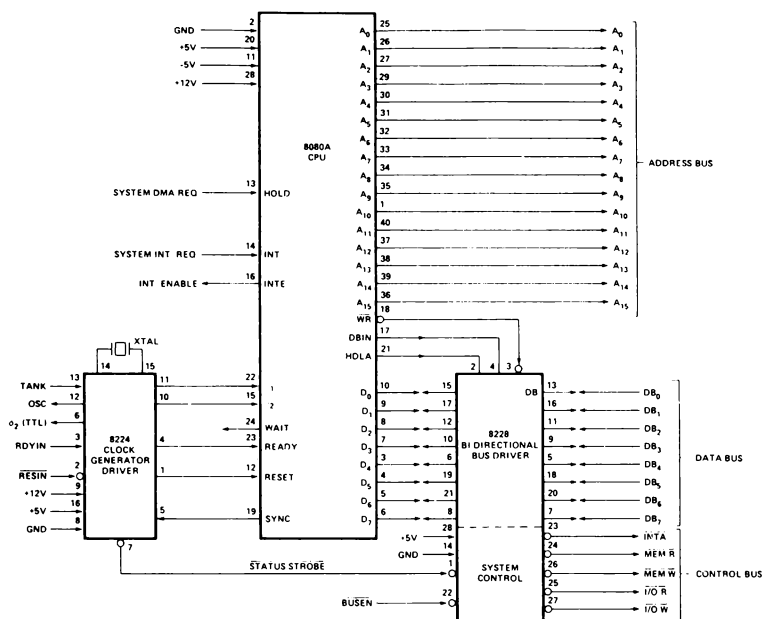



Fig. 6-16. Schema di un tipico circuito di controllo 8080 basato su di un chip 8228.

troller e bus driver 8228, che contiene un latch di stato e i gate necessari all'interno del chip per generare i segnali di controllo $\overline{\text{INTA}}$, $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{I/O R}}$, e $\overline{\text{I/O W}}$, come mostra la Fig. 6-16. Il chip 8228 agisce anche come driver del bus bidirezionale, che permette al bus di dati bidirezionale di essere collegato a più carichi TTL di quanto non possa l'8080 da solo. Pur sostituendo molti chip TTL, l'8228 è ancora relativamente costoso. Ve ne parliamo solo per richiamare la vostra attenzione su questo chip e su altri nuovi chip vi aiutano a semplificare il vostro compito di progettazione del microcomputer. La maggior parte di voi saranno utenti, piuttosto che progettisti, che iniziano con il chip della CPU e vanno avanti. Questo capitolo vi dà un'idea di ciò che la progettazione richiede per convertire un microprocessore 8080 in un microcomputer completo. Se siete interessati alla progettazione dei microcomputer usando i chip dei microprocessori, seguite attentamente le pubblicazioni periodiche e tutto quanto viene scritto dai costruttori. Può darsi che altri abbiano già incontrato e risolto i vostri stessi problemi.

Timing di Stato

Ogni diverso ciclo macchina è suddiviso in *stati* di 500 ns che vanno da tre a cinque e che sono indicati con T_1, T_2, T_3, T_4, T_5. Tre stati addizionali — WAIT, HOLD, HALT — durano da tre ad un numero indefinito di periodi di clock di 500 ns, essendo controllati da segnali esterni che determinano il momento in cui terminano tali stati. Il numero totale di stati per ogni istruzione è determinato dal numero di cicli macchina e dal numero di stati in ogni ciclo macchina. Le istruzioni più semplici dell'8080 hanno quattro stati; le più complicate ne hanno diciotto. Il chip dell'8080 non indica il suo stato interno in modo diretto, ma trasmettendo un'uscita di «controllo di stato» durante ogni stato; l'8080 fornisce uscite di controllo diretto (INTE, HLDA, DBIN, $\overline{\text{WR}}$, WAIT) per l'uso da parte di circuiti esterni.[8]

L'8080 passa attraverso almeno tre stati in ogni ciclo macchina, con ogni stato definito da successive transizioni del fronte principale positivo del clock $\varnothing_1$. Ogni stato contiene due impulsi di clock: un impulso $\varnothing_1$ e un impulso $\varnothing_2$. Gli eventi che accadono in ogni stato si riferiscono a transizioni negli impulsi di clock $\varnothing_1$ e $\varnothing_2$; tali eventi di solito si verificano al fronte principale o entro 50 ns del fronte principale.

Le attività, o azioni, che avvengono durante ogni stato, spiegate nei Riferimenti 8 e 9, si possono riassumere nel modo seguente:

Stato	*Attività associate*
T_1	L'indirizzo di memoria proveniente dal contatore di programma, la coppia di registri H e L, o lo stack pointer, o un numero di dispositivi di I/O, viene posto sul bus degli indirizzi di memoria a 16 bit vicino al fronte principale positivo dell'impulso di clock $\varnothing_2$.
	Il pin di uscita SYNC sul microprocessore 8080 va a livello logico 1 poco dopo il fronte principale positivo dell'impulso di clock $\varnothing_2$.
	Gli otto bit di stato (INTA, $\overline{\text{WO}}$, STACK, HLTA, OUT, M, INP, MEMR) sono posti sul bus di dati bidirezionale esterno poco dopo il fronte principale positivo dell'impulso di clock $\varnothing_2$.
T_2	Gli stati logici dei pin d'ingresso HOLD e READY del chip 8080 e la presenza di un'istruzione HALT vengono provati. Se l'ingresso READY è a livello logico 0, si può inserire lo stato T_3; se l'ingresso READY è a livello logico 0, la CPU va in stato di attesa, T_W.
	Il pin di uscita SYNC sul chip 8080 va a livello logico 0 poco dopo il fronte principale positivo dell'impulso di clock $\varnothing_2$.

Poco dopo il fronte principale positivo dell'impulso di clock $\varnothing_2$, la parola di stato a 8 bit sul bus di dati esterno viene sostituita o da un'istruzione a 8 bit o da un byte di dati provenienti dalla memoria, o da un byte di dati provenienti dall'accumulatore o da un dispositivo d'ingresso.

I contenuti del bus degli indirizzi di memoria a 16 bit non cambiano durante questo stato. DBIN sul chip 8080 va a livello logico 1 al fronte positivo $\varnothing_2$.

T_W Uno stato di attesa opzionale. La CPU va in questo stato se l'ingresso READY del chip 8080 è a livello logico 0 o se è stata eseguita un'istruzione HALT. Se si incontra una istruzione HALT, la CPU resterà in questo stato finché non viene ricevuta un'interruzione o non viene resettato il contatore di programma. La CPU riconosce lo stato di attesa mettendo il pin d'uscita WAIT sul chip 8080 a livello logico 1, al fronte iniziale positivo dell'impulso di clock $\varnothing_1$.

Lo stato di HOLD è un po' più complesso e non ne parleremo qui.

Un periodo di attesa può avere una durata indefinita. Il processore resta nella condizione di attesa finché la sua linea d'ingresso READY va di nuovo a livello logico 1. Può quindi procedere il ciclo istruzioni, iniziando dal fronte principale positivo dell'impulso di clock $\varnothing_1$ seguente. Un'intervallo di WAIT consisterà quindi di un numero intero di stati T_W e sarà sempre un multiplo del periodo di clock.

I contenuti del bus degli indirizzi di memoria a 16 bit non cambiano durante questo stato a meno che la CPU sia nello stato di HOLD.

T_3 I dati provenienti dalla memoria e presenti sul bus di dati esterno a 8 bit possono essere trasferiti nel registro istruzioni, nell'accumulatore o in uno dei registri universali.

I dati provenienti da un dispositivo d'ingresso e presenti sul bus di dati esterno possono essere trasferiti nell'accumulatore.

I dati provenienti dall'accumulatore e presenti sul bus di dati esterno possono essere trasferiti in memoria o in un dispositivo di uscita.

I dati provenienti da uno dei rgistri universali e presenti

sul bus di dati esterno possono essere trasferiti in memoria.

Il pin di uscita DBIN sul chip 8080 ritorna a livello logico 0 al fronte iniziale positivo dell'impulso di clock $\varnothing_2$.

Il pin di uscita $\overline{WR}$ sul chip 8080 va a livello logico 0 al fronte iniziale positivo del primo impulso di clock che segue lo stato T_2.

Notate che questo potrebbe avvenire o nello stato T_W o nello stato T_3, di solito in quest'ultimo.

Durante lo stato T_3, possono svolgersi varie attività, a seconda del tipo di ciclo macchina. Riassumendo, un byte istruzioni (ciclo macchina di fetch), un byte di dati (lettura in memoria, lettura nello stack, o ciclo macchina di ingresso), o un'istruzione d'interruzione (ciclo d'interruzione) viene inserita nella CPU dal bus di dati esterno a 8 bit; oppure un byte di dati (scrittura in memoria, scrittura nello stack, o ciclo macchina di uscita) viene messo in uscita sul bus di dati esterno.[8]

T_4, T_5 Due stati opzionali disponibili per l'esecuzione di una particolare istruzione, in caso di richiesta. In caso contrario, la CPU può saltarne uno o tutte e due. Questi stati si usano solo per le operazioni interne dei processori. I contenuti del bus degli indirizzi di memoria cambiano poco dopo il fronte iniziale positivo dell'impulso di clock $\varnothing_2$ in questi stati.

T_1 Il pin di uscita $\overline{WR}$ sul chip 8080 ritorna a livello logico 1 al fronte iniziale positivo dell'impulso di clock $\varnothing_1$ durante questo stato.
Questo è il primo stato di un nuovo ciclo istruzioni.

Le descrizioni suddette sono riassunte nelle Figg. 6-17 e 6-18. La Fig. 6-17 è un ciclo macchina a cinque stati, il ciclo di fetch, in cui un byte di memoria viene trasferito nel registro istruzioni. Fino a che il pin di uscita READY è a livello logico 1 durante lo stato T_2, si può inserire lo stato T_3. Non abbiamo precisato che cosa succede durante gli stati T_4 e T_5. La Fig. 6-18 è un ciclo macchina a tre stati in cui un byte di dati viene scritto in memoria o in un dispositivo di uscita. Notate che il pin di uscita $\overline{WR}$ sul chip 8080 è a livello logico 0 durante lo stato T_3. DBIN resta a livello logico 0 durante un ciclo macchina di scrittura o di uscita.

Questo pone termine alla nostra discussione sul «timing di stato» nel microprocessore 8080. Varrebbe la pena che studiaste le Figg. 6-11 e 6-12 e vi procuraste anche delle copie dei Riferimenti 8 e 9, che scendono molto nei particolari trattando l'argomento.

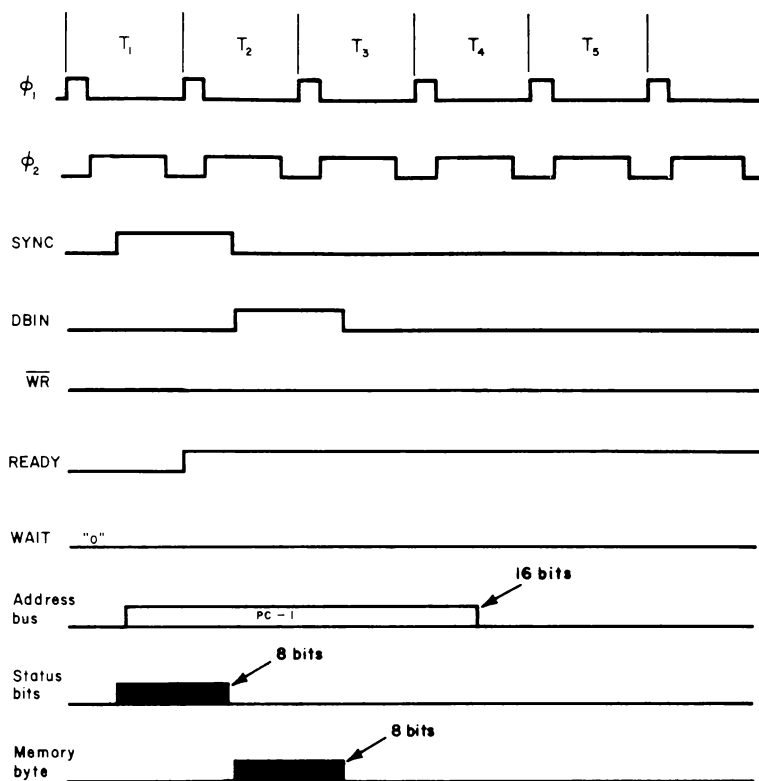



Fig. 6-17. Un ciclo macchina a cinque stati, il ciclo di fetch, in cui un byte di memoria viene trasferito nel registro istruzioni.

FUNZIONAMENTO PASSO-PASSO DI UN MICROCOMPUTER 8080

La Fig. 6-19 mostra un circuito usato con un microcomputer 8080 per procedere all'esecuzione di un programma secondo la tecnica single-step (passo-passo). Viene descritto perché vi permetterà di approfondire la conoscenza dello stato di WAIT del microcomputer 8080.

La funzione principale del circuito della Fig. 6-19 è quella di generare, tramite un generatore d'impulsi con logica antirimbalzo, sul pannello frontale del microcomputer, un impulso monostabile di 550 ns all'ingresso D del flip-flop Positive edge triggered 7474. Come mostra la Fig. 6-20, l'ingresso di clock TTL $\varnothing_1$ del flip-flop 7474 permette l'esistenza di un livello logico 1 al pin 23, l'ingresso READY, del microprocessore 8080. Un livello logico 1 esiste solo per 500 ns, ma questo periodo è sufficiente per permettere al ciclo macchina di passare allo stato T_3 dalla serie di stati di attesa T_W. Oltre allo stato T_3 vi sono lo stato T_4 o forse lo stato T_5, o ancora un nuovo ciclo macchina, che inizia con lo stato T_1. In altre paro-

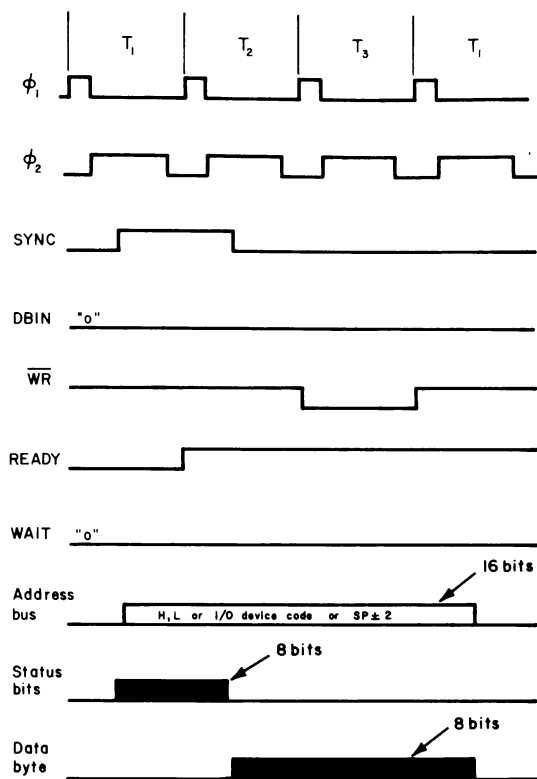



Fig. 6-18. Un ciclo macchina a tre stati in cui un byte di dati viene scritto in memoria o in un dispositivo di uscita. Notate che $\overline{WR}$ è a livello logico 0 durante lo stato T_3.

le, il generatore di impulsi passo-passo sul pannello frontale permette al ciclo macchina di inserire lo stato T_3 da uno stato di attesa. Notate che l'uscita WAIT sul chip 8080 va a livello logico 0 all'inizio dello stato T_3.

Lo switch con logica antirimbalzo vi permette di disabilitare il circuito generatore d'impulsi, attuato secondo la tecnica del passo-passo. Nel suo stato quiescente, lo switch con logica antirimbalzo applica un livello logico 0 al pin 2 del gate NAND 7400. Questo costringe l'uscita del gate ad essere a livello logico 1, uno stato logico che è temporizzato attraverso il flip-flop 7474 al pin 23 del chip 8080. Finché l'ingresso READY del chip 8080 è a livello logico 1, non è possibile attuare la tecnica del passo-passo nel programma.

Notate che il generatore d'impulsi passo-passo ha come stato quiescente, lo stato logico 1. Questo vi permette di usare un clock esterno per temporizzare il monostabile 74123 al pin 2. Gli autori

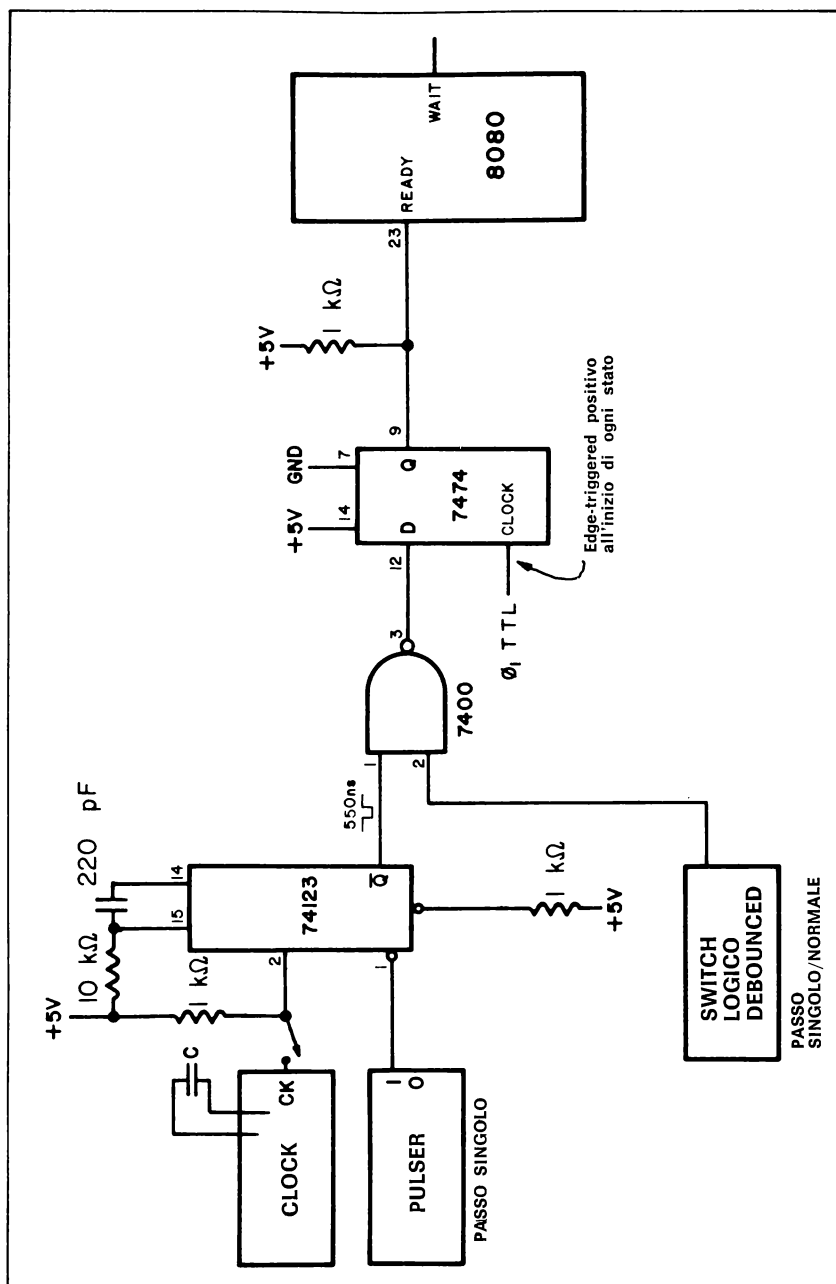



Fig. 6-19. Il circuito impiegato in un microcomputer 8080 per permettere l'esecuzione di un programma secondo la tecnica del passo-passo. La frequenza di clock è di 2 MHz.

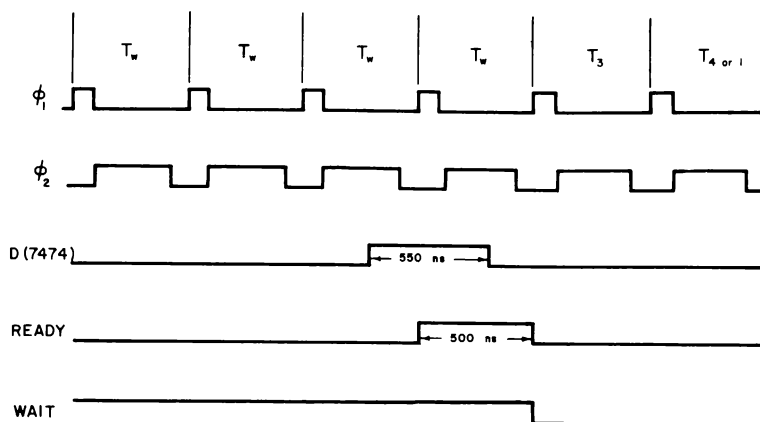



Fig. 6-20. Diagrammi di timing per il circuito che attua la tecnica del passo-passo mostrato nella Fig. 6-19.

hanno verificato che un clock esterno, che consiste di un timer 555 con una frequenza di clock di circa 1 Hz, può essere veramente utile.

C'è un'ultima domanda sul circuito che attua la tecnica del passo-passo: a quale velocità il microcomputer basato sull'8080 esegue una sola istruzione quando viene attuata tale tecnica? La risposta è: alla massima frequenza interna di clock di 2 MHz. Con la capacità del passo-passo, costringete semplicemente il computer a restare nello stato di attesa. Quando portate l'ingresso READY a livello logico 1 per 500 ns, permettete al microcomputer di eseguire gli stati T_3, T_4, T_1 e T_2 del ciclo macchina seguente ad una cadenza di temporizzazione (clock rate) di 500 ns per ogni stato. *Non siete in grado di controllare la velocità alla quale viene eseguito un solo stato. Finché il clock interno ha una frequenza di 2*

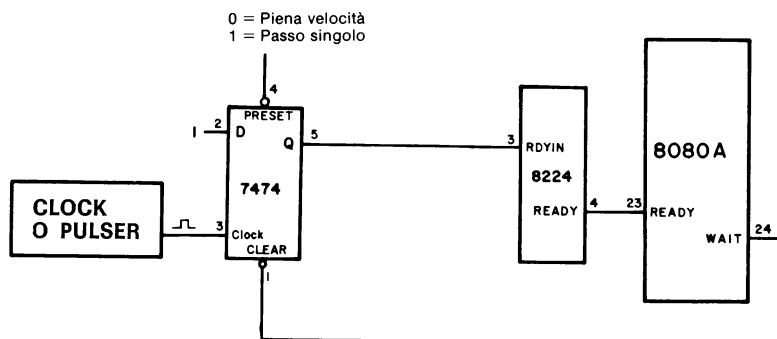



Fig. 6-21. Un semplice circuito con tecnica passo-passo per un microcomputer basato sull'8080 che impiega un chip 8224. Questo circuito può operare a qualunque frequenza di clock del sistema.

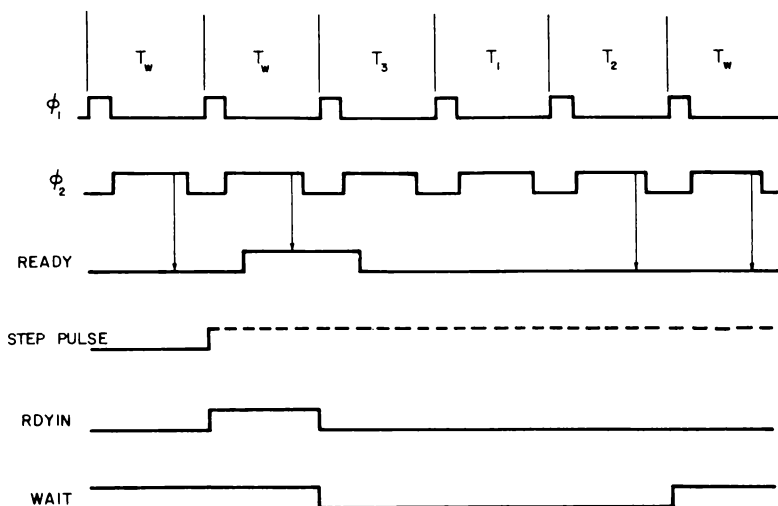



Fig. 6-22. Diagrammi di timing per il circuito dato nella Fig. 6-21. L'impulso a gradino indicato con step pulse, è l'impulso applicato al pin 3 sul chip 7474; può essere generato da un generatore d'impulsi manuale.

MHz, una velocità di questo tipo sarà sempre di 500 ns per ogni stato.

Un secondo e più interessante circuito che attua la tecnica del passo-passo, basato sull'uso di un flip-flop 7474 nei sistemi 8080 che contengono un clock generator/driver 8224, è mostrato nella Fig. 6-21. Potrete capire il comportamento di questo circuito con l'aiuto dei diagrammi di timing mostrati nella Fig. 6-22.

L'operatività del circuito è come quella del circuito precedente; e si basa sul fatto che un livello logico 0 all'ingresso READY costringerà il chip a stare in uno stato di attesa. Come si può vedere dallo schema a blocchi del chip 8224, il segnale di clock $\emptyset_2$ temporizza un latch tipo D entro l'8224, che effettua un latch sull'ingresso RDYN e lo mette in uscita sull'ingresso READY dell'8080. Ecco la sequenza di azioni che hanno luogo nel circuito che attua la tecnica del passo-passo:

- Un impulso di clock, «step pulse», applicato al pin 3 del flip-flop 7474, produce un livello logico 1 all'uscita Q, che è collegata all'ingresso RDYIN del chip 8224.
- Al fronte positivo di $\emptyset_2$, l'uscita READY del chip 8224 va a livello logico 1.
- Il chip 8080 campiona la linea d'ingresso READY durante $\emptyset_2$; con l'ingresso READY ora a livello logico 1, l'8080 porta a termine l'esecuzione del ciclo macchina in corso.
- Alla fine del secondo stato di attesa nel diagramma di tim-

ing, la linea di uscita WAIT del chip 8080 ritorna a livello logico 0, ed azzera il flip-flop 7474.

- RDYIN ritorna al livello logico 0, che viene sottoposto a clock nel chip 8224 all'uscita READY al pin 4.
- Dopo lo stato T_2 del ciclo macchina seguente, il microprocessore 8080 entra di nuovo in uno stato di attesa finché l'ingresso READY è di nuovo a livello logico 0. Se RDYIN viene mantenuto a livello logico 1, il che può succedere se l'ingresso di preset del flip-flop 7474 viene mantenuto a livello logico 0, READY sarà sempre a livello logico 1 e sarà impossibile per il chip 8080 entrare in uno stato di attesa. Gli autori sono grati a Mr. William Dalton, studioso di scienza del computer al VPI e SU, per aver messo in rilievo l'utilità di questo circuito, compreso il fatto che esso non richiede un circuito di timing *RC* e può così operare a qualunque frequenza di clock del sistema. Nelle pagine seguenti vengono fornite le specifiche per il clock generator/driver 8224.

IL CHIP 8212, PORTA DI I/O AD 8 BIT

Il circuito integrato di I/O a otto bit 8212, è destinato ad essere un buffer/latch molto comune, per cui vale la pena di capire come funziona. Fondamentalmente, il chip è un gruppo di otto latch di tipo 7475 ognuno dei quali ha un buffer di uscita three-state. (Fig. 6-23).

Forse ricorderete che l'uscita del latch della Fig. 6-23 segue l'ingresso dei dati finché l'ingresso di clock è a livello logico 1. Un livello logico 0 all'ingresso $\overline{CLR}$ azzera il latch. L'ingresso di abilitazione, EN, del buffer three-state deve essere a livello logico 1, perché i dati appaiano all'uscita del buffer.

Una parte importante dell'8212 è dedicata alla logica di controllo. Lo schema mostrato nella Fig. 6-24 faciliterà la comprensione dei circuiti. I cinque ingressi di controllo si possono così riassumere:

$\overline{DS1}$, DS2 — Selezione dispositivo. Questi due ingressi si usano per la selezione del dispositivo. Quando $\overline{DS1}$ è a livello logico 0 e DS2 è a livello logico 1, il dispositivo è selezionato. Nello stato selezionato, il buffer di uscita viene abilitato e il flip-flop di servizio richiesta (SR) è settato a livello logico 1. Quando l'ingresso MD è a livello logico 1, la sorgente degli impulsi di clock degli otto latch proviene dagli ingressi di selezione dispositivo. Quando l'ingresso MD è a livello logico 0, lo stato del buffer di uscita è determinato dagli ingressi di selezione dispositivo.

Schottky Bipolar 8224

CLOCK GENERATOR AND DRIVER FOR 8080A CPU

- **Single Chip Clock Generator/Driver for 8080A CPU**
- **Power-Up Reset for CPU**
- **Ready Synchronizing Flip-Flop**
- **Advanced Status Strobe**
- **Oscillator Output for External System Timing**
- **Crystal Controlled for Stable System Operation**
- **Reduces System Package Count**

The 8224 is a single chip clock generator/driver for the 8080A CPU. It is controlled by a crystal, selected by the designer, to meet a variety of system speed requirements.

Also included are circuits to provide power-up reset, advance status strobe and synchronization of ready.

The 8224 provides the designer with a significant reduction of packages used to generate clocks and timing for 8080A.

PIN CONFIGURATION

BLOCK DIAGRAM

PIN NAMES

RESIN	RESET INPUT
RESET	RESET OUTPUT
RDYIN	READY INPUT
READY	READY OUTPUT
SYNC	SYNC INPUT
STSTB	STATUS STB (ACTIVE LOW)
φ1	8080
φ2	CLOCKS

XTAL 1	CONNECTIONS
XTAL 2	FOR CRYSTAL
TANK	USED WITH OVERTONE XTAL
OSC	OSCILLATOR OUTPUT
φ2 (TTL)	φ2 CLK (TTL LEVEL)
Vcc	+5V
VDD	+12V
GND	0V

HOTTKY BIPOLAR 8224

FUNCTIONAL DESCRIPTION

General

The 8224 is a single chip Clock Generator/Driver for the 8080A CPU. It contains a crystal-controlled oscillator, a "divide by nine" counter, two high-level drivers and several auxiliary logic functions.

Oscillator

The oscillator circuit derives its basic operating frequency from an external, series resonant, fundamental mode crystal. Two inputs are provided for the crystal connections (XTAL1, XTAL2).

The selection of the external crystal frequency depends mainly on the speed at which the 8080A is to be run at. Basically, the oscillator operates at 9 times the desired processor speed.

A simple formula to guide the crystal selection is:

$$\text{Crystal Frequency} = \frac{1}{t_{CY}} \text{ times } 9$$

Example 1: (500ns t_{CY})
2mHz times 9 = 18mHz*

Example 2: (800ns t_{CY})
1.25mHz times 9 = 11.25mHz

Another input to the oscillator is TANK. This input allows the use overtone mode crystals. This type of crystal generally has much lower "gain" than the fundamental type so an external LC network is necessary to provide the additional "gain" for proper oscillator operation. The external LC network is connected to the TANK input and is AC coupled to ground. See Figure 4.

The formula for the LC network is:

$$F = \frac{1}{2\pi\sqrt{LC}}$$

The output of the oscillator is buffered and brought out on OSC (pin 12) so that other system timing signals can be derived from this stable, crystal-controlled source.

*When using crystals above 10mHz a small amount of frequency "trimming" may be necessary. The addition of a small capacitance (3pF - 10pF) in series with the crystal will accomplish this function.

Clock Generator

The Clock Generator consists of a synchronous "divide by nine" counter and the associated decode gating to create the waveforms of the two 8080A clocks and auxiliary timing signals.

The waveforms generated by the decode gating follow a simple 2-5-2 digital pattern. See Figure 2. The clocks generated; phase 1 and phase 2, can best be thought of as consisting of "units" based on the oscillator frequency. Assume that one "unit" equals the period of the oscillator frequency. By multiplying the number of "units" that are contained in a pulse width or delay, times the period of the oscillator frequency, the approximate time in nanoseconds can be derived.

The outputs of the clock generator are connected to two high level drivers for direct interface to the 8080A CPU. A TTL level phase 2 is also brought out ϕ_2 (TTL) for external timing purposes. It is especially useful in DMA dependant activities. This signal is used to gate the requesting device onto the bus once the 8080A CPU issues the Hold Acknowledgement (HLDA).

Several other signals are also generated internally so that optimum timing of the auxiliary flip-flops and status strobe ($\overline{STSTB}$) is achieved.

Fig. 6-23. Tipico elemento buffer/latch nel circuito integrato 8212. Vi sono otto di questi elementi sul chip.

MD — Modo. Questo ingresso si usa per controllare lo stato del buffer di uscita e per determinare la sorgente dell'ingresso di clock degli otto latch. Quando MD è a livello logico 1, i buffer di uscita sono abilitati e la sorgente degli impulsi di clock proviene dagli ingressi di selezione dispositivo. Quando MD è a livello logico 0, lo stato del buffer di uscita è determinato dagli ingressi di selezione dispositivo e la sorgente degli impulsi di clock degli otto latch proviene dall'ingresso STB.

STB — Strobe. Questo ingresso temporizza gli otto latch quando l'ingresso MD è a livello logico 0. Questo ingresso azzera inoltre, in modo asincrono, il flip-flop di servizio richiesta a Q = 0.

$\overline{\text{CLR}}$ — Azzeramento. Un livello logico 0 a questo ingresso azzera in modo asincrono gli otto latch e setta in modo asincrono il flip-flop di servizio richiesto a Q = 1. Quando il flip-flop di servizio richiesta è settato, esso è nello stato di non-interruzione.

Le tre uscite che risultano dai cinque ingressi di controllo suddetti si possono così descrivere:

$\overline{\text{INT}}$ — Un livello logico 0 a questa uscita si può usare per interrompere il microcomputer. L'uscita deve essere invertita e poi collegata al pin 14 del chip 8080. *Appare come un pin di uscita esterno.*

Clock — Un livello logico 1 abilita gli otto latch tipo D, che seguono l'ingresso dei dati. Il latch viene effettuato quando questa uscita ritorna ad uno stato di livello logico 0. *Avviene internamente al chip.*

Enable — Un livello logico 1 abilita gli otto buffer di uscita three-state, uno su ogni latch. Un livello logico 0 costringe

i buffer a stare nel loro stato di alta impedenza. *Avviene internamente al chip.*

La Tabella 6-2 riassume il comportamento di tutti i circuiti delle Fig. 6-23 e 6-24.

Tabella 6-2: Ingressi e Uscite del Chip 8212

Control Inputs								
$\overline{CLR}$	DS2	$\overline{DS1}$	MD	STB	Enable	Clock	Q(SR)	$\overline{INT}$
1	1	⊔	1	1	1	⊓	1	⊔
1	⊓	0	1	1	1	⊓	1	⊔
1	1	⊔	0	1	⊓	1	0	⊔
1	⊓	0	0	1	⊓	1	0	⊔
1	1	0	0	⊓	1	⊓	0	0

La letteratura della Intel Corporation sulla porta di I/O a otto bit 8212 fornisce varie applicazioni per l'uso di questo chip sotto for-

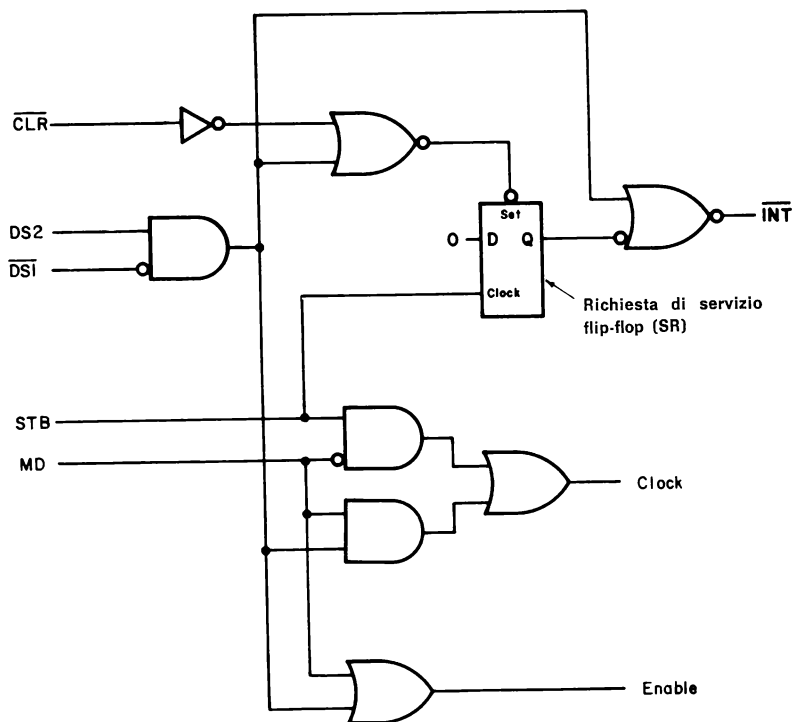



Fig. 6-24. Particolari della logica di controllo e di selezione dispositivo per il circuito integrato 8212. Le uscite di clock e di abilitazione sono collegate direttamente ai latch tipo D 7475 all'interno del chip stesso, come mostra la Fig. 6-23. Non esistono come uscite del chip 8212.

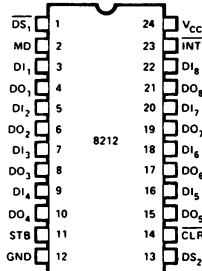
Schottky Bipolar 8212

EIGHT-BIT INPUT/OUTPUT PORT

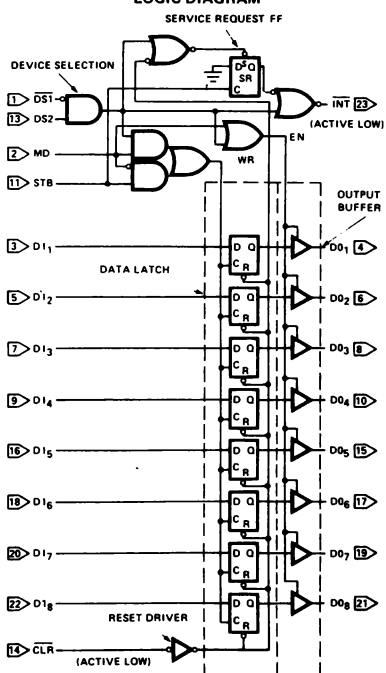
- **Fully Parallel 8-Bit Data Register and Buffer**
- **Service Request Flip-Flop for Interrupt Generation**
- **Low Input Load Current — .25 mA Max.**
- **Three State Outputs**
- **Outputs Sink 15 mA**
- **3.65V Output High Voltage for Direct Interface to 8080 CPU or 8008 CPU**
- **Asynchronous Register Clear**
- **Replaces Buffers, Latches and Multiplexers in Microcomputer Systems**
- **Reduces System Package Count**

The 8212 input/output port consists of an 8-bit latch with 3-state output buffers along with control and device selection logic. Also included is a service request flip-flop for the generation and control of interrupts to the microprocessor.

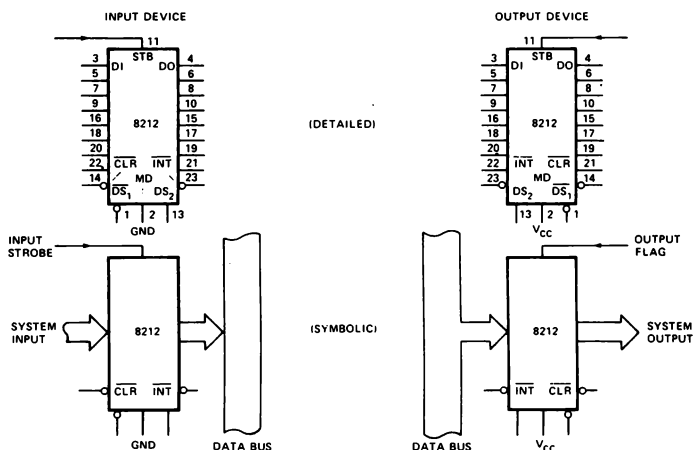
The device is multimode in nature. It can be used to implement latches, gated buffers or multiplexers. Thus, all of the principal peripheral and input/output functions of a microcomputer system can be implemented with this device.

PIN CONFIGURATION

PIN NAMES

DI_1 DI_8	DATA IN
DO_1 DO_8	DATA OUT
$\overline{DS_1}$ DS_2	DEVICE SELECT
MD	MODE
STB	STROBE
$\overline{INT}$	INTERRUPT (ACTIVE LOW)
$\overline{CLR}$	CLEAR (ACTIVE LOW)

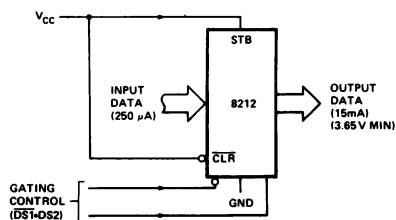
LOGIC DIAGRAM

Applications Of The 8212 -- For Microcomputer Systems

I. Basic Schematic Symbols

Two examples of ways to draw the 8212 on system schematics—(1) the top being the detailed view showing pin numbers, and (2) the bottom being the symbolic view showing the system input or output as a system bus (bus containing 8 parallel lines). The output to the data bus is symbolic in referencing 8 parallel lines.

II. Gated Buffer (3 - STATE)

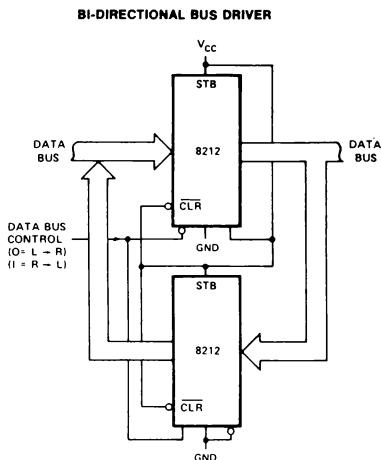
The simplest use of the 8212 is that of a gated buffer. By tying the mode signal low and the strobe input high, the data latch is acting as a straight through gate. The output buffers are then enabled from the device selection logic $\overline{DS1}$ and DS2.

When the device selection logic is false, the outputs are 3-state.

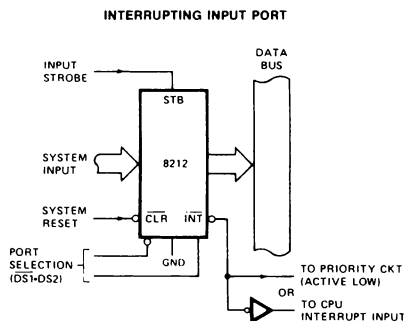
When the device selection logic is true, the input data from the system is directly transferred to the output. The input data load is 250 micro amps. The output data can sink 15 milli amps. The minimum high output is 3.65 volts.

SCHOTTKY BIPOLAR 8212

III. Bi-Directional Bus Driver

A pair of 8212's wired (back-to-back) can be used as a symmetrical drive, bi-directional bus driver. The devices are controlled by the data bus input control which is connected to $\overline{DS1}$ on the first 8212 and to DS2 on the second. One device is active, and acting as a straight through buffer the other is in 3-state mode. This is a very useful circuit in small system design.

BI-DIRECTIONAL BUS DRIVER

IV. Interrupting Input Port

This use of an 8212 is that of a system input port that accepts a strobe from the system input source, which in turn clears the service request flip-flop and interrupts the processor. The processor then goes through a service routine, identifies the port, and causes the device selection logic to go true — enabling the system input data onto the data bus.

INTERRUPTING INPUT PORT

V. Interrupt Instruction Port

The 8212 can be used to gate the interrupt instruction, normally RESTART instructions, onto the data bus. The device is enabled from the interrupt acknowledge signal from the microprocessor and from a port selection signal. This signal is normally tied to ground. ($\overline{DS1}$ could be used to multiplex a variety of interrupt instruction ports onto a common bus).

INTERRUPT INSTRUCTION PORT

SCHOTTKY BIPOLAR 8212

VI. Output Port (With Hand-Shaking)

The 8212 can be used to transmit data from the data bus to a system output. The output strobe could be a hand-shaking signal such as "reception of data" from the device that the system is outputting to. It in turn, can interrupt the system signifying the reception of data. The selection of the port comes from the device selection logic. ($\overline{DS1}$ • DS2)

OUTPUT PORT (WITH HAND-SHAKING)

VII. 8080 Status Latch

Here the 8212 is used as the status latch for an 8080 microcomputer system. The input to the 8212 latch is directly from the 8080 data bus. Timing shows that when the SYNC signal is true, which is connected to the DS2 input and the phase 1 signal is true, which is a TTL level coming from the clock generator; then, the status data will be latched into the 8212.

Note: The mode signal is tied high so that the output on the latch is active and enabled all the time.

It is shown that the two areas of concern are the bidirectional data bus of the microprocessor and the control bus.

8080 STATUS LATCH

ma di buffer gated, bus driver bidirezionale, porta d'ingresso di interruzione, porta istruzione d'interruzione, porta di uscita, e latch di stato per l'8080. Nelle pagine seguenti vi presentiamo quattro pagine tratte dalla letteratura della Intel. Il chip è veloce, e la maggior parte delle azioni avvengono in non più di 40 ns.

TEST

Questo test verifica quanto avete capito sui concetti avanzati discussi in questo capitolo. Per favore, scrivete le risposte su un foglio a parte.

6.1 Qual'è la differenza fra un ciclo di clock, un ciclo macchina e un ciclo istruzione?

6.2 Quali tipi di informazioni a 8 bit possono apparire sul bus di dati esterno? Date il maggior numero di particolari possibile.

6.3 Quali fra gli elementi della memorizzazione dati elencati precedentemente sono collegati al bus di dati esterno, e quali al bus di dati interno?

accumulatore
registro B
un byte di memoria di lettura/scrittura
un dispositivo d'ingresso
un dispositivo di uscita
il registro istruzioni
i flag
il byte di memoria PROM.

6.4 Descrivete cinque degli otto bit di stato e spiegate che tipo di informazioni essi forniscono.

6.5 Descrivete cinque dei nove cicli macchina.

6.6 Sia i bit di stato che i pin di uscita del microprocessore 8080 si possono usare per fornire segnali di uscita per interfacciare i circuiti. Spiegate come questo avviene.

6.7 Descrivete i diversi tipi di stati che possono comprendere un ciclo macchina.

Le vostre risposte saranno accettabili se sarete in grado di rispondere a tutte le domande in modo corretto, a libro chiuso e in 90 minuti di tempo. I concetti suddetti sono concetti avanzati che, probabilmente, non avrete bisogno di conoscere subito. Col tempo, comunque, capirete che sono molto importanti nello sviluppo di circuiti di interfaccia più complessi.

CHE COSA AVETE REALIZZATO IN QUESTO CAPITOLO?

All'inizio di questo capitolo, era stato stabilito che, alla fine, avreste dovuto essere in grado di:

- Spiegare le differenze fra il bus di dati interno ed esterno in un microcomputer 8080.

  Questo è stato fatto nella parte riguardante il bus di dati bidirezionale. Il data sheet delle specifiche tecniche sul microprocessore 8080 fornito dalla Intel, dà un'eccellente visione del bus di dati interno.

- Fare un elenco delle sorgenti e delle destinazioni di informazioni che appaiono sul bus di dati esterno.

  Anche questo è stato fatto nella parte riguardante il bus di dati bidirezionale.

- Spiegare le differenze fra uno stato, un ciclo di clock, un ciclo istruzione, e un ciclo macchina.

  Uno stato e un ciclo di clock sono la stessa cosa. Un ciclo macchina consiste di tre o più stati. Un ciclo istruzione consiste di cicli macchina da uno a cinque. Tutti questi concetti sono stati trattati dettagliatamente.

- Spiegare che cosa sono un bit di stato e un byte di stato.

  Un byte di stato consiste di otto diversi bit di stato, di cui abbiamo parlato nella parte riguardante l'identificazione dei cicli macchina.

- Descrivere i nove tipi diversi di cicli macchina.

  Essi sono fetch, lettura in memoria, scrittura in memoria, uscita, ingresso, scrittura nello stack, lettura nello stack, alt, interruzione. Questi nove tipi di cicli sono trattati, nei minimi particolari, nella parte riguardante i cicli macchina.

- Descrivere la funzione di ognuno degli otto diversi bit di stato.

  Gli otto bit di stato sono presentati nella parte riguardante l'identificazione dei cicli macchina. Vengono usati per identificare il tipo di ciclo macchina che è in fase di esecuzione.
  La Fig. 6-21 fornisce il circuito che si può usare con un microcomputer basato sull'8080.

- Spiegare in che modo le uscite di controllo sul chip 8080 si possono combinare in modo logico con uno o più bit di stato, e fornire almeno un esempio di una combinazione logica di questo tipo.

  Nel testo abbiamo combinato in modo logico i seguenti segnali:

  OUT e $\overline{WR}$
  INP e DBIN
  M_1 e DBIN

  La prima coppia generava un impulso $\overline{OUT}$ di 500 ns, la seconda coppia generava un impulso $\overline{IN}$ di 500 ns, e la terza generava un singolo impulso di clock all'inizio di ogni ciclo istruzione.

- Disegnare diagrammi di timing che illustrino il comportamento delle istruzioni tipiche del microcomputer. Tali diagrammi dovrebbero dimostrare chiaramente gli stati logici degli ingressi di controllo, delle uscite di controllo e dei bit di stato più importanti.

  Ciò è stato fatto dettagliatamente in questo capitolo, vedi Fig. 6-7.

- Spiegare i diversi tipi di dati che appaiono sul bus di dati esterno.

  Qui vengono messi in rilievo i dati, non i byte istruzione. I dati possono provenire dall'accumulatore, dalla memoria, da un dispositivo d'ingresso, dal secondo e dal terzo byte di un'istruzione, e da un registro.

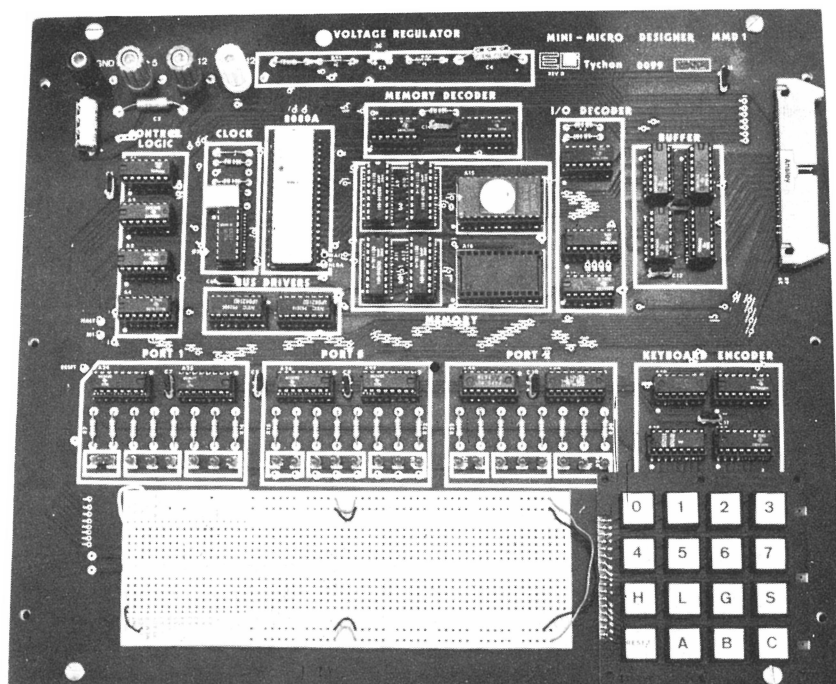
- Spiegare il timing di stato per cicli macchina tipici.

  Lo abbiamo fatto nel sottoparagrafo riguardante il timing di stato. Abbiamo parlato di sei diversi stati. Le Figg. 6-17 e 6-18 forniscono degli esempi.

- Spiegare come viene attuata la tecnica del passo-passo in un microcomputer basato sull'8080, completa di diagrammi di timing e di schemi di circuito.

  La Fig. 6-21 fornisce il circuito che si può usare con un microcomputer basato sull'8080.

- Discutere delle caratteristiche del circuito integrato 8212, porta di ingresso/uscita a otto bit.

  Lo abbiamo fatto alla fine del capitolo. Abbiamo fornito del materiale scritto dai costruttori, in proposito.

Mini-Micro Designer MMD-1 della E & L Instruments

CAPITOLO 7

Input / Output del Microcomputer

In questo capitolo, imparerete come usare gli impulsi di selezione dispositivo per effettuare un latch sui dati provenienti dall'accumulatore e per inserire i dati nell'accumulatore. Verranno forniti dei circuiti per sei diversi latch e due diversi buffer d'ingresso. Imparerete anche come provare varie istruzioni relative all'accumulatore, compresa l'istruzione di aggiustamento decimale dell'accumulatore stesso, che vi permette di eseguire operazioni su dati in BCD.

OBIETTIVI

Alla fine di questo capitolo, sarete in grado di:

- Effettuare un latch sull'uscita di sei diversi circuiti integrati: 7475, 74100, 74175, 75193, 74198, 8212.
- Inserire dati TTL nell'accumulatore con l'aiuto del buffer 8095 o del buffer 8212.
- Spiegare che cos'è un registratore automatico di dati (data logger).
- Calcolare i ritardi di timing richiesti per registrare automaticamente le informazioni digitali che appaiono a velocità diverse.

DEFINIZIONI

Accumulator (Accumulatore) Il registro e l'insieme dei circuiti elettronici digitali ad esso associati nell'unità aritmetica di un computer in cui vengono eseguite operazioni logiche e aritmetiche.

Autoranging instrument (Strumento ad autorange) Uno strumento digitale che modifica automaticamente il suo range operativo.

Buffer Un dispositivo digitale che isola un circuito digitale dall'altro. (Nota: Esiste una varietà di altri significati del termine «buffer»).

Bus monitor Un dispositivo che serve a controllare i segnali

	digitali che appaiono su di un bus.
Data point	Tutti i bit necessari per caratterizzare il segno e la grandezza di una quantità digitale misurata. Un data point consiste di solito di molti bit.
Decimal adjust accumulator (Aggiustamento decimale dell'accumulatore)	Un'istruzione del microprocessore 8080 che permette le operazioni su dati decimali codificati in binario.
Data logger (Sistema di acquisizione dati)	Uno strumento che scandisce automaticamente i dati prodotti da un altro strumento o elabora e registra le letture dei dati, da utilizzarsi in un secondo tempo.
Input/output (Ingresso/uscita)	Termine generale per le apparecchiature usate per comunicare con un computer e con i dati coinvolti nella comunicazione.[4]
I/O	Abbreviazione di input/output.[4]
Latch	Un semplice elemento di memoria logica.
Log (Registrare)	Scandire automaticamente i dati prodotti da uno strumento o elaborare e registrare le letture dei dati da utilizzarsi in un secondo tempo.
Monitor	Un dispositivo che si usa per controllare i segnali.[4]
Rotate (Rotazione)	Una istruzione che fa sì che i contenuti dell'accumulatore si spostino, bit per bit, verso sinistra o verso destra, di una posizione.

INGRESSO/USCITA

Quando si usa il termine *ingresso/uscita*, o *I/O*, quello che si intende dire di solito è che uno o più byte di dati vengono trasferiti fra un dispositivo di I/O e il microcomputer. Nelle Figg. 7-1 e 7-2 potete vedere una coppia di diagrammi che mostrano in che modo questo avviene. I punti importanti che possiamo rilevare in queste due figure si possono così riassumere:

- L'ingresso/uscita di un byte di dati avviene fra un dispositivo esterno di I/O e l'*accumulatore* all'interno del chip 8080.
- Gli impulsi $\overline{IN}$ e $\overline{OUT}$ sono prodotti dal microprocessore 8080 (con l'ausilio di tutti i circuiti addizionali, come descritto nel Capitolo 6) per permettere ad un dispositivo di I/O di sapere se esso sta trasferendo un byte di dati nel microcomputer o se sta ricevendo un byte di dati.
- Un codice dispositivo a 8 bit disponibile nel momento in cui viene generato un impulso $\overline{IN}$ o $\overline{OUT}$, permette di generare 256 impulsi di selezione dispositivi d'ingresso e di 256 diversi impulsi di selezione dispositivi di uscita.

In altre parole, parti di informazioni digitali vengono trasferite da un punto all'altro all'interno di un sistema a microcomputer interfacciato con il mondo esterno, durante un ciclo delle istruzioni IN o OUT.

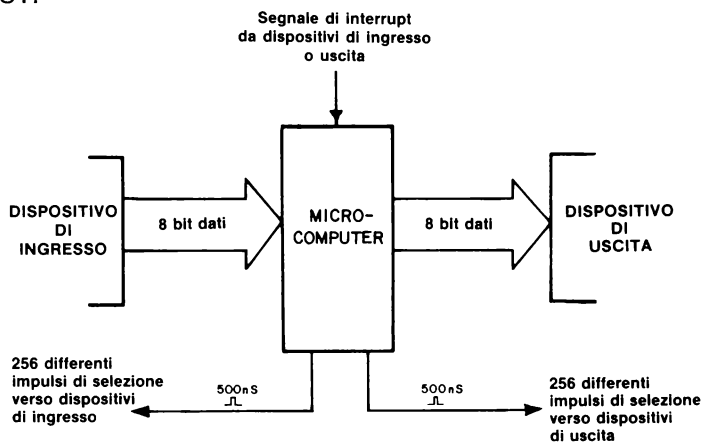



Fig. 7-1. Gli ingressi e le uscite importanti di un tipico sistema di microcomputer basato sull'8080.

CIRCUITI DI USCITA DA UN MICROCOMPUTER

L'espediente di base da adottare per ottenere l'uscita dei dati

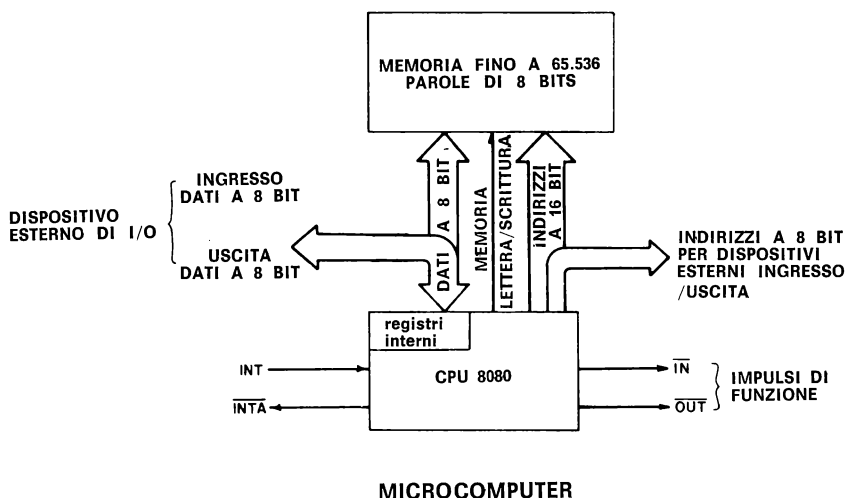



Fig. 7-2. Tipici ingressi ed uscite di un microprocessore 8080. Gli impulsi $\overline{\text{IN}}$ e $\overline{\text{OUT}}$ sono generati con l'aiuto di circuiteria addizionale, cioè di un latch 8212.

dall'accumulatore è molto semplice: *usate un solo impulso di selezione dispositivo di uscita per abilitare un latch che agisca su di un byte di dati proveniente dal bus di dati bidirezionale.* I dati entrano ed escono sulle stesse linee del chip 8080: da D0 a D7. A seconda del tipo di latch, si usa un impulso di selezione dispositivo positivo o negativo, per fornire impulsi di strobe, semplici impulsi, o per effettuare un latch sui dati. Gli impulsi di selezione dispositivo sono prodotti con l'aiuto di un decodificatore, come il 7442 o il 74154, in cui viene effettuato il gate tra il bus degli indirizzi ed i segnali di controllo $\overline{\text{IN}}$ o $\overline{\text{OUT}}$. Basandovi sulle informazioni fornite nel Capitolo 4, dovreste essere in grado di disegnare vari schemi di decodifica e di gating per ricavare gli impulsi di selezione dispositivo d'ingresso e di uscita.

I circuiti di uscita tipici dei microcomputer sono quelli basati sul chip 8212 (Figg. 7-3A e 7-3B), il latch tipo D 74100 a otto bit (Figg. 7-4A e 7-4B), una coppia di latch tipo D 7475 (Figg. 7-5A e 7-5B), il registro a scorrimento a otto bit 74198 e una coppia di latch positive edge triggered 74175 (Figg. 7-6A, 7-6B e 7-6C), ed anche una coppia di contatori up/down 74193 (Figg. 7-7A e 7-7B).

Quando usate il chip 8212 come latch di uscita, dovete assicurarvi che l'ingresso di azzeramento, $\overline{\text{CLR}}$, al pin 14 sia collegato al livello logico 1. Questo ingresso, normalmente, dovrebbe spostarsi al livello logico 1 quando non è collegato, ma l'esperienza dimostra che non è sempre così.

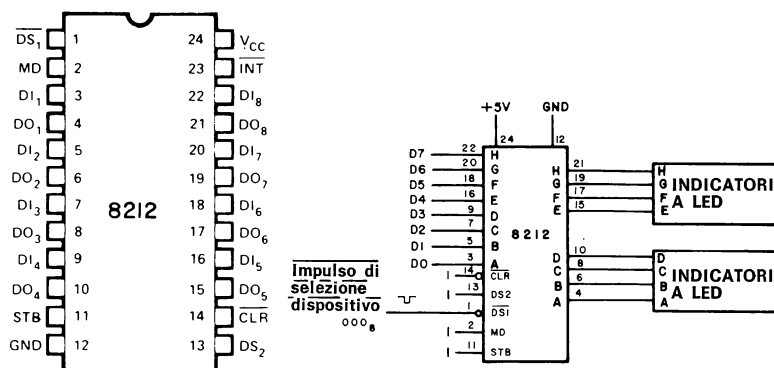



Fig. 7-3. Circuito di uscita del microcomputer basato sul chip 8212.

CIRCUITI D'INGRESSO DI UN MICROCOMPUTER

L'espediente di base da adottare per inserire i dati nell'accumulatore da un dispositivo d'ingresso è ugualmente semplice: *usate un solo impulso di selezione dispositivo d'ingresso per abilitare un buffer three-state, che trasferisce un byte di dati nel bus di dati bidirezionale.* Potete usare un buffer/latch three-state come l'8212 (Fig. 7-8), o una coppia di buffer three-state 8095 (74365), che so-

no relativamente economici (Figg. 7-9A e 7-9B).

In questo caso, i buffer abilitati permettono ai dati di essere trasferiti nel bus di dati bidirezionale, da D0 a D7, cioè dal pin 3 al pin 10 sul chip 8080. L'accumulatore acquisisce i dati durante i 500 ns dell'impulso di selezione dispositivo d'ingresso. Nelle illustrazioni precedenti, l'impulso di selezione dispositivo è un impulso di clock negativo, come indicano i trattini sulla parola «Impulso di selezione dispositivo».

Gli ingressi del buffer three-state 8095 sono collegati agli switch logici, ma possono essere collegati a qualunque sorgente di otto bit di dati TTL binari o codificati. Questi dati vengono trasferiti attraverso i buffer 8095, posti sul bus di dati, e copiati nell'accumulatore durante un'istruzione IN del microcomputer. I dati vengono inseriti ogni volta che viene eseguita l'istruzione IN.

ISTRUZIONI DI INGRESSO/USCITA

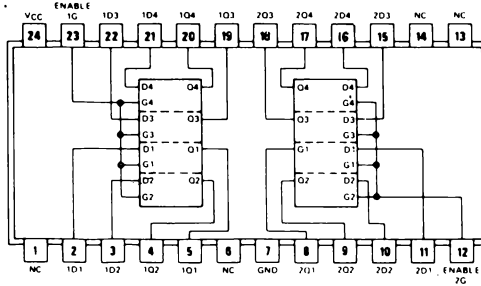
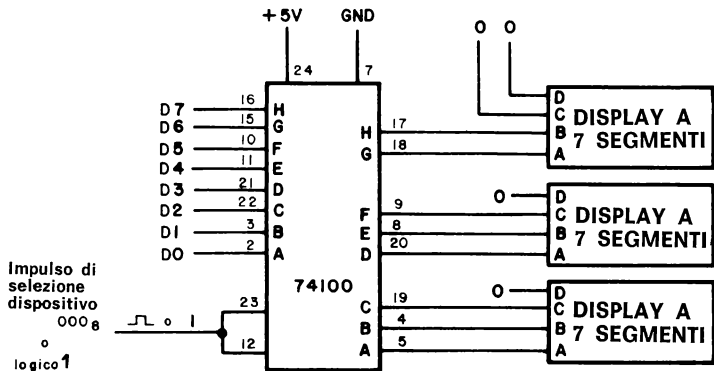
Ripeteremo le due istruzioni di I/O del microprocessore 8080 date nel Capitolo 4:

(A) Configurazione dei pin del latch a otto bit 74100.

(B) Circuito in cui un chip 74100 funge da latch di uscita. L'uscita è fornita sotto forma di parola ottale a tre cifre.

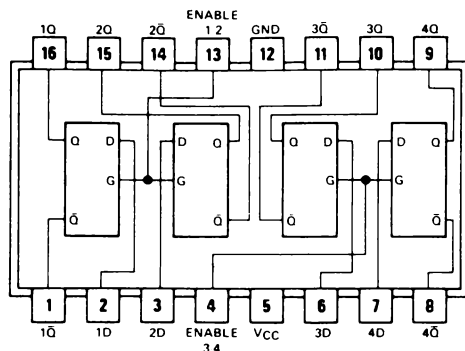
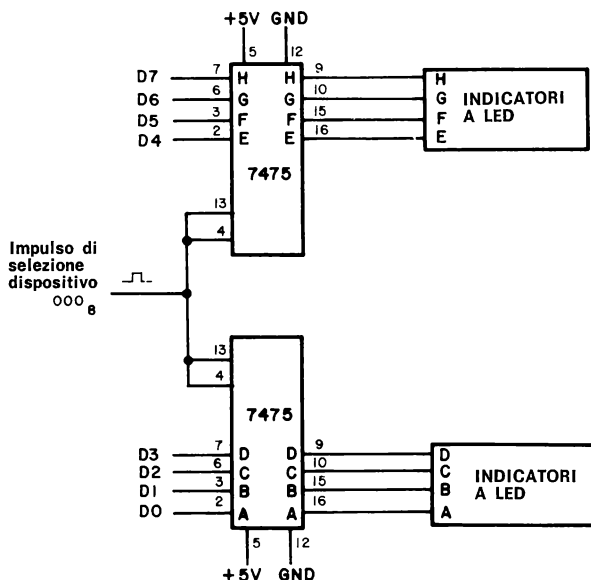
Fig. 7 4. Circuito di uscita del microcomputer basato sul latch 74100.

(A) Configurazione dei pin del latch a quattro bit 7475.

(B) Circuito in cui una coppia di latch tipo D 7475 funge da porta di uscita per il microcomputer.

Fig. 7-5. Circuito di uscita del microcomputer basato su di un latch 7475.

333 <B2> IN — Genera un impulso di selezione dispositivo d'ingresso per permettere ad un byte di dati a 8 bit di essere letto da un dispositivo d'ingresso e per sostituire i contenuti dell'accumulatore.

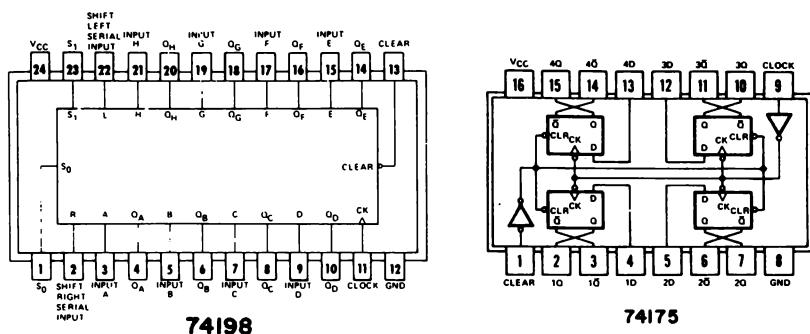
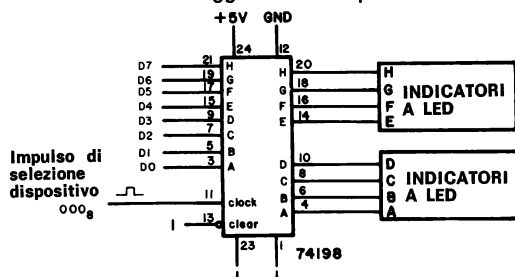
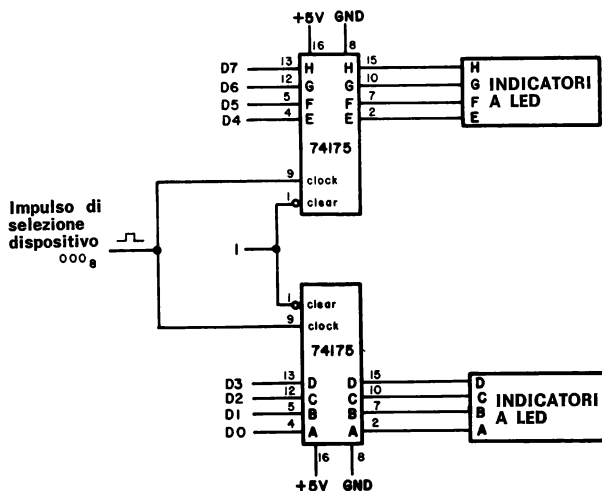
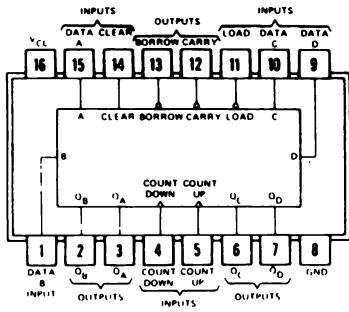
323 <B2> OUT — Genera un impulso di selezione dispositivo di uscita per fare sì che un byte di dati a 8 bit presente nell'accumulatore venga inviato ad un dispositivo di uscita. I contenuti dell'accumulatore restano invariati.

(A) Configurazione dei pin del registro a scorrimento a otto bit 74198 e del latch a quattro bit 74175. Entrambi i chip contengono flip-flop positive edge triggered del tipo 7474.

(B) Circuito in cui un registro a scorrimento 74198 effettua un latch sui dati dell'accumulatore.

(C) Circuito in cui una coppia di latch 74175 effettua un latch sui dati dell'accumulatore a 8 bit.

Fig. 7-6. Circuiti di uscita del microcomputer basati sul chip 74198 e 74175.

logic: Low input to load sets $Q_A = A$, $Q_B = B$, $Q_C = C$, and $Q_D = D$

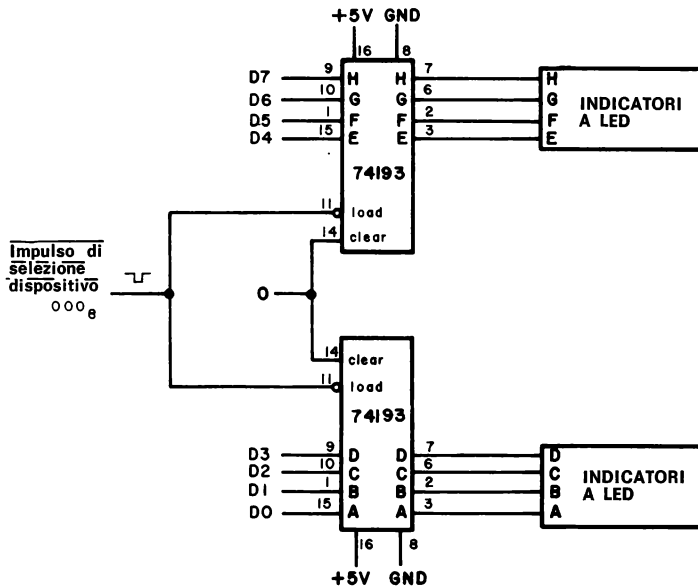
(A) Il contatore up/down 74192 o 74193. Questo chip contiene un latch a quattro bit del tipo 7475.

(B) Circuito latch di uscita che impiega un paio di contatori up/down 74193. Il microcomputer viene usato per conteggio in questi contatori.

Fig. 7-7. Circuito di uscita del microcomputer basato su di un chip 74193.

Il secondo byte di ognuna delle istruzioni, <B2>, è un codice dispositivo a 8 bit che vi permette di selezionare uno qualunque dei 256 diversi dispositivi d'ingresso o dei 256 diversi dispositivi di uscita.

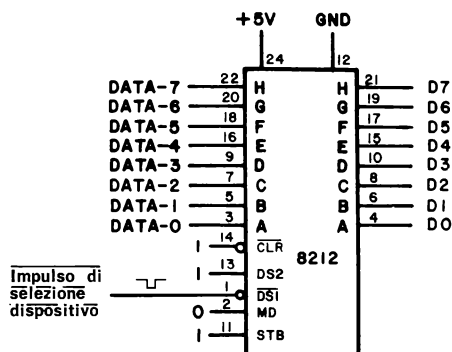



Fig. 7-8. Circuito in cui un chip 8212 viene usato sotto forma di buffer d'ingresso del microcomputer 8080.

PROGRAMMI DI INGRESSO/USCITA

Vi mostriamo un semplice programma per inserire i dati provenienti dallo switch logico della Fig. 7-9B nell'accumulatore e poi metterli automaticamente in uscita in uno dei circuiti di uscita del microcomputer, mostrati in precedenza:

Indirizzo di memoria LO	*Istruzione ottale*	*Codice mnemonico*	*Commento*
000	333	IN	Inserisci i dati dal dispositivo d'ingresso 004
001	004	004	Codice dispositivo 004
002	323	OUT	Poni in uscita i dati sul dispositivo di uscita 000
003	000	000	Codice dispositivo 000
004	166	HLT	Alt

Questo programma inserisce dati dallo switch logico, li mette in uscita su un latch di uscita e poi si arresta. Per inserire e mettere in uscita i dati continuamente, dovreste cambiare il programma in questo modo:

Indirizzo di memoria LO	*Istruzione ottale*	*Codice mnemonico*	*Commento*
000	333	IN	Inserisci i dati dal dispositivo d'ingresso 004
001	004	004	Codice dispositivo 004
002	323	OUT	Poni in uscita i dati sul dispositivo di uscita 000
003	000	000	Codice dispositivo 000
004	303	JMP	Salto incondizionato alla locazione di memoria data dai due byte seguenti
005	000		Byte d'indirizzo LO
006	000		Byte d'indirizzo HI

Per memorizzare i dati in ingresso in una locazione di memoria ed

aggiornare i contenuti della memoria ogni volta che vengono inseriti nuovi dati, dovreste usare il programma seguente:

Indirizzo di memoria LO	*Istruzione ottale*	*Codice mnemonico*	*Commento*
000	333	IN	Inserisci i dati dal dispositivo d'ingresso 004
001	004	004	Codice dispositivo 004
002	323	OUT	Poni in uscita i dati sul' dispositivo di uscita 000
003	000	000	Codice dispositivo 000
004	062	STA	Memorizza i contenuti dell'accumulatore nella locazione di memoria data dai seguenti due byte
005	200		Byte d'indirizzo LO
006	003		Byte d'indirizzo HI
007	303	JMP	Salto incondizionato alla locazione di memoria data dai due byte seguenti
010	000		Byte d'indirizzo LO
011	000		Byte d'indirizzo HI

Il programma è simile a quello mostrato in precedenza in questo paragrafo, ma questa volta è stata aggiunta un'istruzione STA <B2> <B3> per permettervi di memorizzare i contenuti dell'accumulatore in una specifica locazione di memoria, i cui contenuti vengono aggiornati durante ogni loop di programma. Potreste chiedervi: «Come possono venire memorizzati i dati quando sono stati inviati in precedenza al display, che nell'esempio del programma precedente possiede il codice dispositivo 000?. I dati non vengono «consumati» quando sono messi in uscita su di un dispositivo? La risposta è no. Quando un byte di dati viene trasferito da una locazione ad un'altra, viene *copiato* nella nuova locazione. Esso non si è consumato; i dati originali sono ancora presenti nella locazione iniziale, sia essa l'accumulatore, un registro, o una locazione di memoria. I dati che si trovano in una data locazione, come ad esempio nell'accumulatore, possono essere copiati indefinitamente.

Mentre il concetto dell'uso di impulsi di selezione dispositivo d'ingresso per inserire otto bit di dati dallo switch logico è diretto, una volta che avete inserito i dati, potete eseguire degli interessanti giochetti di programmazione per avvantaggiarsi della versatilità dell'8080. Per esempio, supponiamo che gli otto bit di dati degli switch logici siano realmente il codice ASCII derivato da una tastiera ASCII standard, con uscita TTL. Ogni volta che vengono inseriti otto nuovi bit di dati, essi vengono testati per determinare se essi sono o non sono l'equivalente ASCII della lettera E, che ha un codice ASCII di 305. Se così, i dati in ingresso vanno posti in uscita e anche memorizzati se no il diagramma di flusso applicabile a questo programma è fornito nella Fig. 7-10 ed il programma è:

Indirizzo di memoria LO	*Istruzione ottale*	*Codice mnemonico*	*Commento*
000	333	IN	Inserisci i dati dal dispositivo d'ingresso 004
001	004	004	Codice dispositivo 004
002	376	CPI	Confronta i contenuti dell'accumulatore con il byte di dati seguente. Se sono uguali, setta il flag di zero
003	305	305	Byte di dati, codice ASCII per la lettera E
004	302	JNZ	Salta alla locazione di memoria data dai due byte seguenti, se il flag di zero è resettato, cioè a livello logico 0
005	000		Byte d'indirizzo LO
006	000		Byte d'indirizzo HI
007	323	OUT	Poni in uscita i dati sul dispositivo di uscita 000
010	000	000	Codice dispositivo 000
011	062	STA	Memorizza i contenuti dell'accumulatore nella locazione di memoria data dai due byte seguenti
012	200		Byte d'indirizzo LO
013	003		Byte d'indirizzo HI
014	166	HLT	Alt

L'istruzione immediata di confronto agli indirizzi di memoria LO 002 e 003 vi permette di confrontare il byte ASCII 305 con i contenuti dell'accumulatore *senza alterare i contenuti dell'accumulatore stesso.* Cambiano solo i contenuti dei flag. Se il byte ASCII per la lettera E e i contenuti dell'accumulatore sono identici, il flag di zero è settato a livello logico 1; altrimenti, viene resettato a livello logico 0. Lo stato del flag viene poi usato nella istruzione seguente di salto condizionato, JNZ, per decidere se continuare ad effettuare il loop o se continuare passando all'istruzione OUT, all'indirizzo di memoria LO di 007.

USCITA DI UN MICROCOMPUTER VERSO UN DISPLAY SOTTOPOSTO A MULTIPLEXER

Una caratteristica importante dei microcomputer è la loro abilità di sostituire l'hardware, cioè i circuiti integrati, con il software. Lo potete vedere bene dall'esempio precedente, in cui un programma verificava l'ingresso del codice ASCII equivalente alla lettera E. Semplici modifiche dei precedenti programmi di I/O cambiano del tutto la natura di quello che ha fatto il microcomputer. Questi stessi cambiamenti occuperebbero molto più tempo per essere sviluppati e provati se fossero eseguiti solo con l'hardware.

Un display sottoposto a multiplexer è un buon esempio di tradeoff hardware-software. Supponiamo che vogliate visualizzare fino a cinque cifre decimali su di un display. Il classico sistema hardware sarebbe quello di usare un latch, un driver di decodifica, ed un display a sette segmenti per ogni numero. Questo procedi-

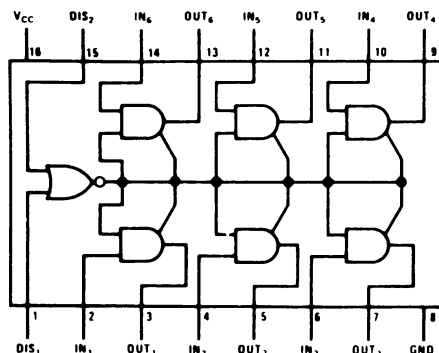
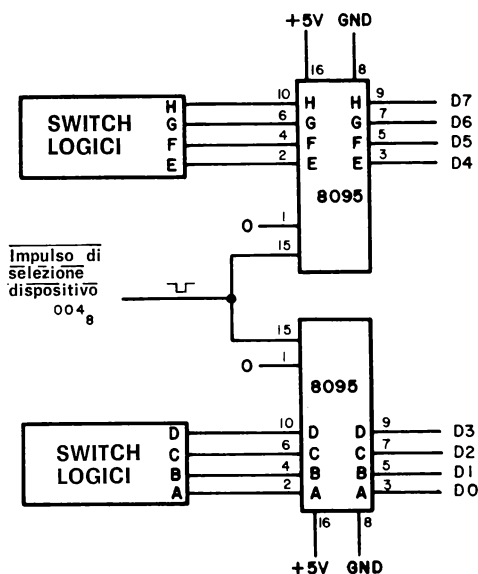
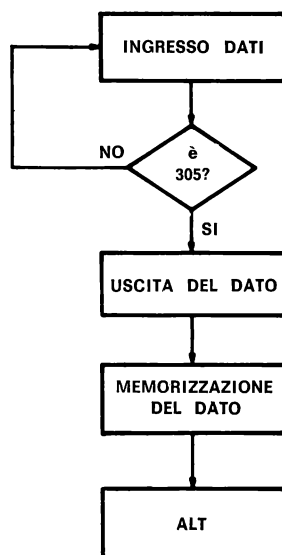
mento può essere costoso ed inoltre può consumare parecchio. Nella Fig. 7-11 vi mostriamo un sistema alternativo, che richiede sia l'hardware che il software. Usate il microcomputer per scandire le cifre molto velocemente. Così facendo, ad una velocità sufficiente, sembra a chi guarda che ogni cifra appaia in modo continuo, come ci si aspetterebbe per un normale display. Le caratteristiche principali del circuito sono un latch 74175 per memorizza-

(A) Il buffer three-state 8095. Viene effettuato un gate su tutti e sei i buffer three-state contemporaneamente, e vi è un gate NOR a due ingressi agli ingressi di abilitazione.

(B) Circuito d'ingresso che impiega due buffer three-state 8095. Gli switch logici possono essere sostituiti da qualunque sorgente a 8 bit di dati TTL.

Fig. 7-9. Circuito d'ingresso del microcomputer, basato sul chip 8095.

re temporaneamente i dati che devono essere visualizzati, un decodificatore/driver 7448 per decodificare questi dati in un'uscita per ognuno dei sette segmenti e diodi luminosi (LED) sul display, e un decodificatore 74154 per selezionare le cifre individuali nel display a cinque cifre. Il 7448 codifica i segmenti propri che devono essere visualizzati nello stesso momento in cui il decodificatore 74154 sceglie una sola cifra. In pratica, potrebbe essere usato un decodificatore «high current», come il 74145, al posto del 74154. Il programma per il display è diretto. Non è necessario che il computer prenda delle decisioni, per cui non vi sono salti condizionati.

Fig. 7-10. Diagramma di flusso del programma che verifica un carattere d'ingresso per determinare se è o meno il carattere ASCII E. Quando viene rilevata finalmente una E, il programma la mette in uscita, la memorizza e poi si arresta.

Indirizzo di memoria LO	*Istruzione ottale*	*Commento*
000	227	Azzera l'accumulatore
001	323	Genera un impulso di selezione dispositivo che si può usare per delle altre operazioni, come l'azzeramento del latch 74175 (il cablaggio non si vede nel grafico)
002	005	Codice dispositivo per altre operazioni
003	041	Carica i due byte seguenti nei registri L e H, rispettivamente
004	200	Byte del registro L
005	001	Byte del registro H
006	176	Trasferisci i contenuti della locazione di memoria indirizzata dalla coppia di registri H e L nell'accumulatore
007	323	Genera un impulso di selezione dispositivo che applichi un livello logico 0 al pin 7 del display a cinque cifre Hewlett-Packard
010	000	Codice dispositivo per il pin 7 del display
011	043	Incrementa i contenuti della coppia di registri H ed L di uno

012	176	Trasferisci i contenuti della locazione di memoria indirizzata dalla coppia di registri H e L nell'accumulatore (NOTA: la coppia di registri è stata incre-
013	323	mentata di 1 all'indirizzo di memoria 006) Genera un impulso di selezione dispositivo che applichi un livello logico 0 al pin 4 del display a cinque cifre Hewlett Packard
014	002	Codice dispositivo per il pin 4 del display Incrementa i contenuti della coppia di registri H ed L di uno
015	043	Genera un impulso di selezione dispositivo che
016	176	applichi un livello logico 0 al pin 13 del display a cinque cifre Hewlett Packard
017	323	Trasferisci i contenuti della posizione di memoria indirizzata dalla coppia di registri H e L nell'accumulatore
020	003	Codice dispositivo per il pin 13 del display
021	043	Incrementa i contenuti della coppia di registri H ed L di uno
022	176	Trasferisci i contenuti della posizione di memoria indirizzata dalla coppia di registri H e L nell'accumulatore
023	323	Genera un impulso di selezione dispositivo che applichi un livello logico 0 al pin 1 del display a cinque cifre Hewlett Packard
024	044	Codice dispositivo per il pin 1 del display
025	076	Trasferisci il byte seguente nell'accumulatore
026	017	Questo byte, che è inserito nel chip 7448, vuota i sette segmenti del display al pin 9 e lascia apparire solo il punto decimale
027	323	Genera un impulso di selezione dispositivo che applichi un livello logico 0 al pin 9 ed un livello logico al pin 5 del display a cinque cifre Hewlett-Packard
030	001	Codice dispositivo per il pin 9 e il pin 5 del display
031	303	Salto incondizionato alla locazione di memoria data dai due byte seguenti
032	000	Byte d'indirizzo LO
033	000	Byte d'indirizzo HI

I numeri da visualizzare devono essere presenti all'interno dell'accumulatore prima che questo programma venga eseguito. Per esempio, le cifre 7, 3, 0 e 5 sono memorizzate nella locazione di memoria HI = 001 e LO = 200 fino a 203. La posizione del punto decimale viene data dall'hardware, la posizione di blank è generata in software. I numeri vengono messi in uscita da destra a sinistra nel circuito, per cui i dati sono memorizzati in questo modo:

Indirizzo di memoria LO	*Byte di dati*
200	007
201	003
202	000
203	005

Un programma o un segmento di programma eseguito in prece-

denza avrebbe potuto porre questi dati nelle locazioni di memoria indicate. Questi dati potrebbero essere inseriti da qualche altro dispositivo, o potrebbero essere il risultato di operazioni aritmetiche. Oggi, i display sottoposti a multiplexer costituiscono ormai una regola. Quasi tutti i display dei calcolatori sono di questo tipo. Il diagramma di flusso del programma precedente è mostrato nella Fig. 7-12. Notate che vi è un solo loop, e che viene eseguito alla massima velocità.

ACQUISIZIONE DATI CON UN MICROCOMPUTER 8080

Un sistema di acquisizione dati si può così definire:

Data logger (Sistema di acquisizione dati) Uno strumento che scandisce automaticamente i dati prodotti da un altro strumento o elabora e registra le letture di dati da utilizzarsi in un secondo tempo.

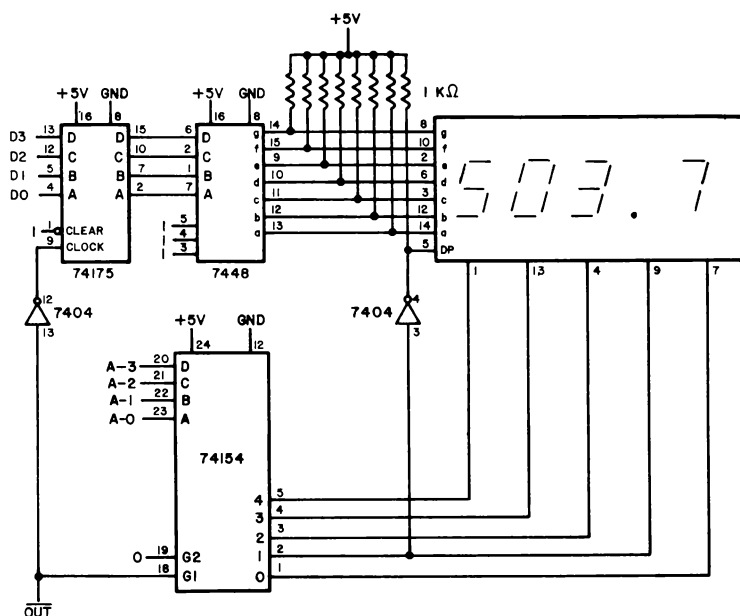



Fig. 7-11. Un display a LED sottoposto a multiplexer, a sette segmenti e a cinque cifre, della Hewlett-Packard.

Dovrebbe essere chiaro il fatto che un microcomputer può costituire un sistema di acquisizione dati. I dati provenienti da uno strumento possono essere messi nell'accumulatore e poi immagazzinati in memoria. In un secondo tempo, queste informazioni memorizzate possono essere lette in uno qualunque dei vari modi esistenti. L'acquisizione dati diventerà un'applicazione comune, nel futuro, per i microcomputer.

Forse le domande più importanti da porvi quando progettate di acquisire i dati da uno strumento sono le seguenti: (1) Quanti canali dati volete acquisire? (2) Quanto tempo ci vorrà per acquisire tutti i canali dati? (3) Quante informazioni digitali sono contenute in un solo canale dati? (4) Che cosa volete farne dei dati, una volta acquisiti? (5) Avrete bisogno della memorizzazione dati a breve o a lungo termine? Risponderemo ora ad ognuna di queste domande.

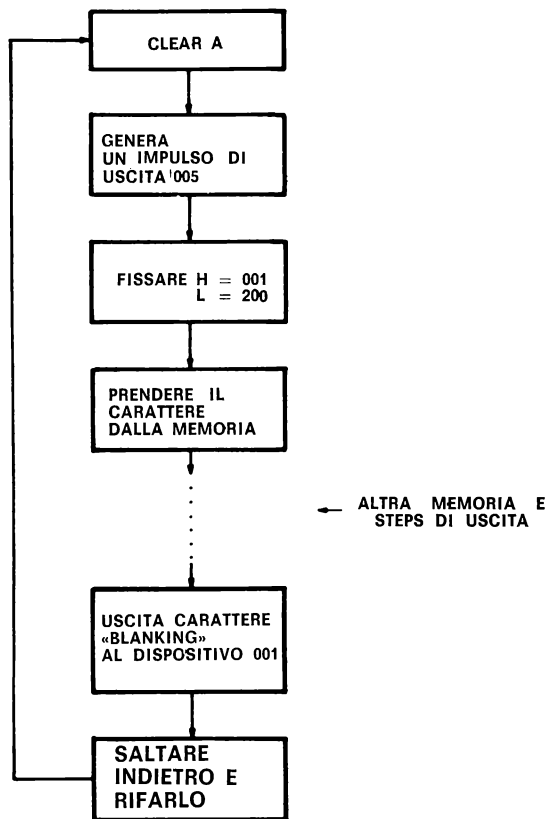



Fig. 7-12. Diagramma di flusso per l'operazion del display a sette segmenti e a quattro cifre della Hewlett-Packard, mostrato nella Fig. 7 11.

Quanti Canali Dati?

Il numero di canali dati che volete acquisire e il tempo che vi occorrerà per memorizzarli indicheranno il tipo di dispositivo di memorizzazione richiesto. Se avete bisogno di acquisire 1.000.000 di parole decimali codificate binarie a quattro cifre, vi servirà una capacità di memoria di sedici milioni di bit. Avrete quindi bisogno di un qualche tipo di nastro o disco magnetico. D'altra parte, se

avete bisogno di acquisire 100 canali, ognuno dei quali contiene cifre bcd, e di memorizzare i dati solo per alcune ore, sono necessari solo 1600 bit di memoria. Una semplice memoria di lettura/scrittura andrà benissimo per un'applicazione di questo tipo.

Memorizzazione a Breve o a Lungo Termine?

Generalmente, la memoria di lettura/scrittura non si adatta alla memorizzazione di dati a lungo termine. Per una ragione: questa memoria è volatile; se c'è un abbassamento di corrente, tutti i dati andranno perduti. La memoria a nuclei non è volatile, ma d'altra parte è molto costosa e generalmente non è adatta per la memorizzazione a lungo termine dei dati a meno che il numero di dati memorizzati sia limitato. Al presente, i migliori dispositivi di memorizzazione dati sembrano essere i nastri magnetici, come una cassetta, e un disco magnetico, come i sempre più comuni dischi «floppy». Una cassetta a nastro di buona qualità può memorizzare fino a 500.000 bit di informazione su di una sola cassetta che costa non più di 15.000 lire.

Una tecnica non costosa e che prevede la memorizzazione a lungo termine è l'uso di nastro di carta perforato. Comunque, non bisogna dimenticare che occorre parecchio tempo per perforare questo nastro nonché per rileggerlo in un computer.

Quante Informazioni in un Solo Canale Dati?

Un canale dati tipico è di solito un numero decimale codificato binario a molte cifre, oppure un numero binario che contiene sia una o più posizioni per le cifre decimali, che un segno. Di solito la/le posizioni delle cifre decimali sono fissate e il segno è positivo, ma non è sempre così. I nuovi dispositivi digitali vanno incorporando sempre di più una capacità di *autoclassificazione,* il che significa semplicemente che il dispositivo decide dove mettere la posizione delle cifre decimali. In generale, potete lavorare su di un canale dati che contenga almeno sedici bit di informazioni digitali. Con 100 canali dati, dovete moltiplicare 100 per il numero di bit di ogni canale dati, 16, per ottenere la capacità complessiva di memoria richiesta, cioè 1600 bit. I frequenzimetri di solito hanno molti più bit per ogni canale dati. Un frequenzimetro a 7 digit ha almeno 28 bit per ogni canale dati.

Che Cosa ne Farete dei Dati Acquisiti?

Alcuni dei dati acquisiti possono essere considerati solo dei dati «rozzi» che devono essere manipolati e interpretati per produrre un risultato finale utile. Per esempio, potrebbe essere necessario convertire una tensione digitale in forza. In casi come questo, i dati acquisiti dovranno essere elaborati matematicamente, il che avverrà meglio subito dopo che i dati sono stati acquisiti. Chiaramente, con dati che richiedono un trattamento addizionale, la cosa migliore è tenere i dati nella forma elettronica digitale

fino a che possono essere manipolati. Sia la memoria di lettura/scrittura che il nastro magnetico vanno benissimo a questo scopo.

Una volta che i dati sono nella loro forma finale, possono essere stampati. Ricordatevi che la stampa dei dati è una forma di memorizzazione a lungo termine. E' sicuramente la forma di memorizzazione a lungo termine meno costosa, ma in compenso dovete perdere del tempo per riconvertirli in un tipo di memorizzazione elettronica o magnetica, se volete eseguire manipolazioni matematiche addizionali sui dati stessi.

Quanti Canali al Secondo?

Questa è una domanda di fondamentale importanza in tutte le operazioni di acquisizione dati. I dati possono, ad esempio:

- Apparire molto lentamente ed occupare dei periodi molto lunghi, come un giorno, per essere acquisiti, o
- Apparire in modo estremamente rapido, ed occupare solo dei millisecondi per l'acquisizione di centinaia di canali dati.

Entrambi gli estremi delle velocità di acquisizione dati necessitano delle tecniche automatiche di acquisizione dati, quali l'uso di un sistema di acquisizione dati basato sul microcomputer. Non c'è dubbio che i dati, sia nei laboratori che da qualunque altra parte, verranno sempre più acquisiti automaticamente da microcomputer incorporati. Si possono ancora usare registratori su carta, ma non sarà più necessario che siano della qualità richiesta negli anni passati. In futuro, un maggiore uso di registratori su carta potrà servire semplicemente per permettere di «integrare» visivamente un blocco di dati per scoprire curvatura, linearità, ecc.

Come dimostrazione dell'acquisizione dati, vorremmo fornire un programma che ci permette di acquisire 256 canali dati a otto bit con la velocità con cui il microcomputer può inserirli e memorizzarli. Per esempio, supponiamo di acquisire dati da una coppia di contatori a decade 7490, come mostra la Fig. 7-13. La domanda a cui cerchiamo di rispondere è questa: Qual'è il tempo minimo richiesto per acquisire 256 canali dati da due contatori, dove ogni canale dati contiene due cifre bcd?

Indirizzo di memoria LO	*Istruzione ottale*	*Cicli di clock*	*Commento*
000	061	10	Carica i due byte seguenti nel registro stack pointer
001	100		Byte dello stack pointer LO
002	003		Byte dello stack pointer HI
003	006	7	Carica il byte di dati seguente nel registro B
004	000		Numero di canali che saranno acquisiti dal microcomputer
005	041	10	Carica i due byte seguenti nella coppia di registri H ed L
006	000		Byte del registro L
007	001		Byte del registro H

010	323	10	Genera un impulso di selezione dispositivo per settare il flip-flop 7476
011	000		Codice dispositivo per l'ingresso di preset
012	333	10	Genera un impulso di selezione dispositivo che permetta a otto bit di dati provenienti dalla coppia di contatori 7479 di essere inseriti nell'accumulatore
013	000		Codice dispositivo per i buffer d'ingresso
014	167	7	Trasferisci i contenuti dell'accumulatore nella locazione di memoria data dalla coppia di registri H ed L
015	043	5	Inserisci la coppia di registri H e L di uno
016	005	5	Decrementa i contenuti del registro B di uno
017	302	10	Se il registro B è 000_8, ignora questa istruzione. Altrimenti, salta all'indirizzo di memoria dato dai due byte seguenti
020	012		Byte d'indirizzo LO
021	000		Byte d'indirizzo HI
022	323	10	Genera un impulso di selezione dispositivo che azzeri il flip-flop 7476
023	001		Codice dispositivo per l'ingresso di azzeramento
024	166		Alt

Nel programma, abbiamo fornito una coppia di istruzioni OUT per poter impiegare le tecniche descritte nel Capitolo 5 per conteggia-

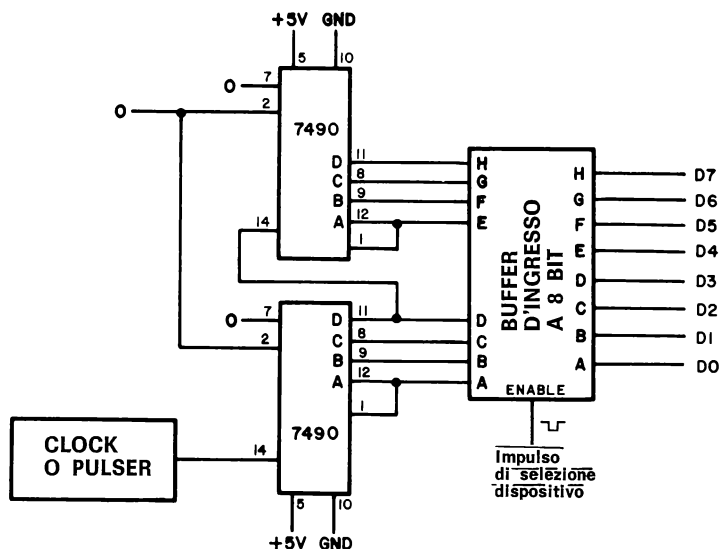



Fig. 7-13. Un semplice circuito di acquisizione dati che impiega una coppia di contatori, in cascata, 7490.

re gli impulsi di clock, come il circuito del flip-flop 7476 mostrato nella Fig. 7-14. L'impulso di selezione dispositivo 000 setta il flip-flop, e l'impulso di selezione dispositivo 001 lo azzera. Mentre è settato, il contatore a 5 decadi conteggia i cicli di clock dal clock del microcomputer.

Sono necessari trentasette cicli di clock per ogni canale dati a 8 bit acquisito dal programma. Per 256 canali dati ed un microcomputer che opera a 2 MHz, sono necessari in tutto 256 x 37 = 9472 cicli di clock, o 4,741 ms. A 18,5 μs per ogni canale dati a 8 bit, si possono acquisire circa 54.000 canali dati al secondo. E questo non è ancora il limite.

Se volete acquisire dati ad una velocità minore, vi servirà una subroutine di delay da inserire nel programma suddetto.

La subroutine di 0,2 secondi descritta nel capitolo 5 dovrebbe andare bene per velocità inferiori. Le modifiche necessarie per il programma suddetto sono piuttosto semplici:

016	016	7	Carica il byte seguente nel registro C
017	*		Byte di timing per la subroutine di delay
020	315	17	Richiama la subroutine all'indirizzo di memoria dato dai due byte seguenti
021	000		Byte d'indirizzo LO
022	060		Byte d'indirizzo HI
023	005	5	Decrementa i contenuti del registro B di uno
024	302	10	Se il registro B è 000_8, ignora questa istruzione Altrimenti, salta all'indirizzo di memoria dato dai due byte seguenti

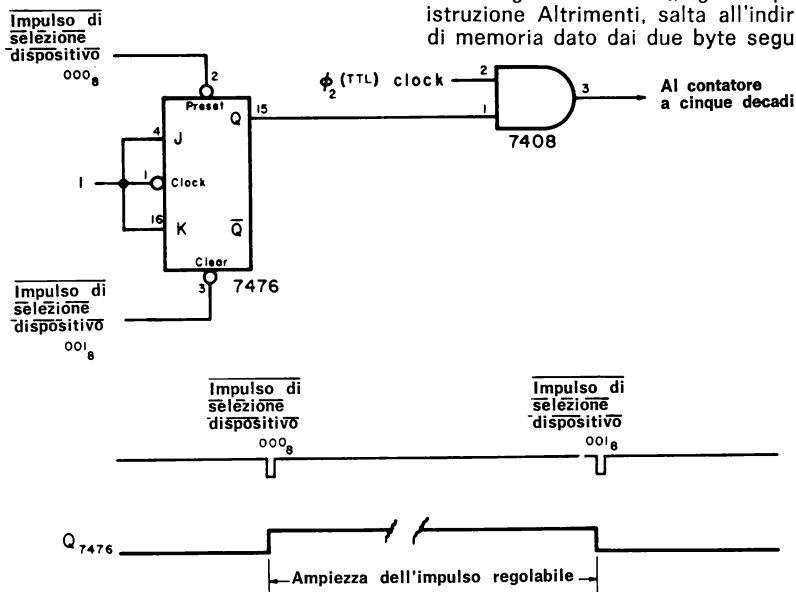



Fig. 7-14. Il circuito latch 7476 vi permette di determinare sperimentalmente quanto ci vuole per acquisire e memorizzare 256 canali dati. Il comportamento del circuito è mostrato nel diagramma di timing.

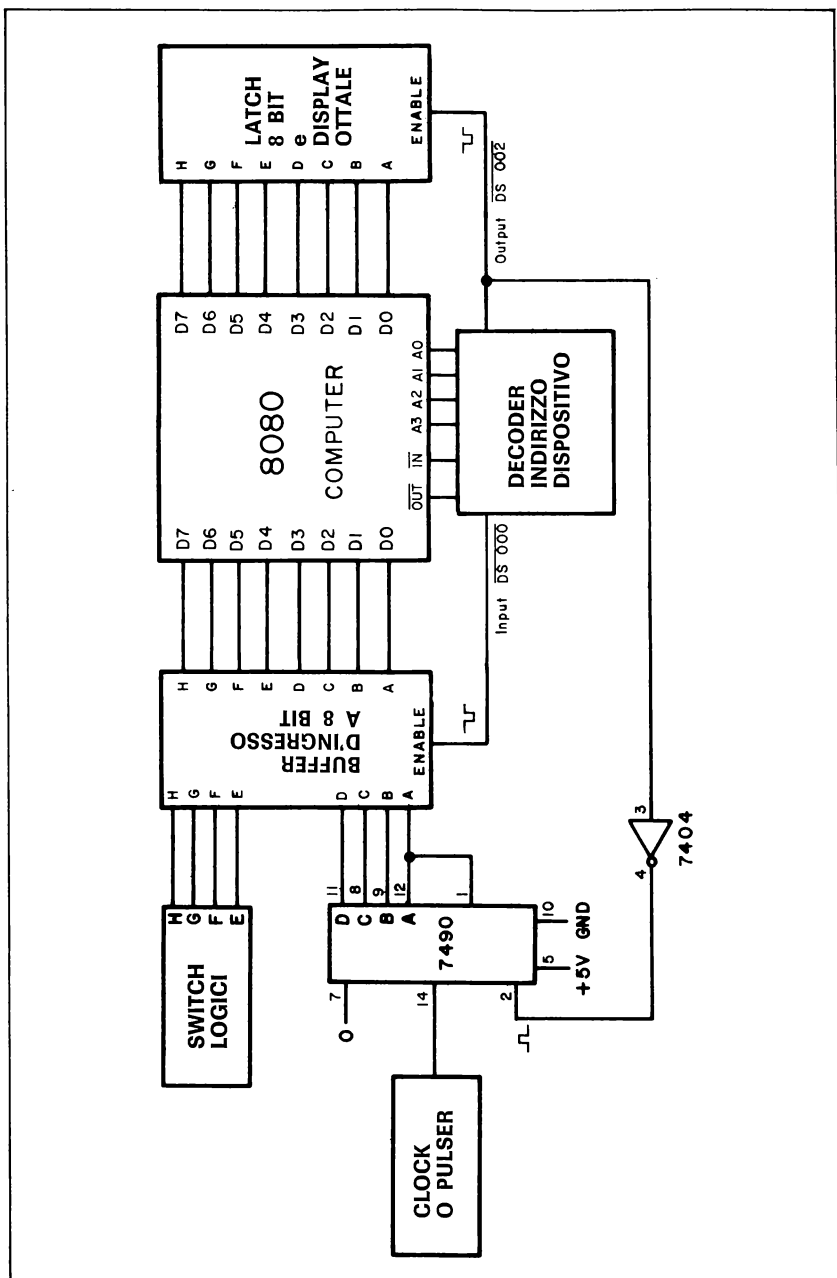



Fig. 7-15. Circuito di acquisizione dati che mette in uscita dati numerici ASCII. Gli switch logici HGFE dovrebbero essere settati a 1011, rispettivamente.

025	012		Byte d'indirizzo LO
026	000		Byte d'indirizzo HI
027	323	10	Genera un impulso di selezione dispositivo che azzeri il flip-flop 7476
030	003		Codice dispositivo
031	166		Alt

Se il byte di dati all'indirizzo di memoria 017 è settato a 001, il che corrisponde ad un ritardo di 0,20 secondi per ogni canale dati, ci vorranno ora circa 51,3 secondi per acquisire tutti e 256 canali. Cambiando semplicemente il byte di dati all'indirizzo di memoria 017, potete cambiare il tempo per ogni canale dati da 0,20 secondi per canale a 51,2 secondi per canale. Questa capacità mette in rilievo i vantaggi dei microcomputer usati come sistemi di acquisizione dati.

Per mettere in uscita il «file» di 256 canali dati, usate un circuito che assomiglia a quello mostrato nella Fig. 7-15. Il programma che usate per mettere in uscita sequenzialmente i canali dati memorizzati è simile al programma che avete usato inizialmente per acquisirli. Quindi, nel programma precedente di questo paragrafo, apportate i seguenti cambiamenti:

012	176	MOV M, A	7	Trasferisci i contenuti della locazione di memoria data dalla coppia di registri H e L nell'accumulatore
013	323	OUT	10	Poni in uscita i dati dell'accumulatore sul latch a 8 bit e sul display ottale
014	002	002		Codice dispositivo per il latch/display

CHE COSA AVETE REALIZZATO IN QUESTO CAPITOLO?

All'inizio di questo capitolo era stato stabilito che, alla fine, avreste dovuto essere stati in grado di:

- Effettuare un latch sull'uscita dell'accumulatore con l'aiuto di uno qualunque dei sei diversi circuiti integrati: 7475, 74100, 74175, 74193, 74198 e 8212.
  Questo è stato fatto nel paragrafo «Circuiti di uscita di un microcomputer»
- Inserire i dati TTL nell'accumulatore con l'aiuto del buffer 8095 o 8212.
  Questo è stato fatto paragrafo «Circuiti d'ingresso microcomputer».
- Spiegare che cos'è un sistema di acquisizione dati.
  Nel testo è stato tratato l'argomento dei sistemi di acquisizione dati. Abbiamo fornito un circuito e un programma per creare un semplice sistema di acquisizione dati basato sul microcomputer.
- Calcolare i ritardi di timing richiesti per acquisire i dati delle informazioni digitali che appaiono a velocità diverse.
  Di questo abbiamo parlato nel testo.

CAPITOLO 8

Subroutine, Interrupt, Flag Esterni e Stack

In questo capitolo focalizzerete la vostra attenzione principalmente sul quarto ed ultimo aspetto fondamentale dell'interfacciamento di un computer: la gestione dell'interrupt (interruzioni). Abbiamo già parlato di alcuni particolari riguardanti l'argomento nel Capitolo 1. Qui, imparerete come definire un flag esterno e come creare un'interfaccia fra questo flag ed il microcomputer. Di solito, i microcomputer operano usando l'interrupt, quindi i concetti descritti in questo capitolo sono molti importanti. Studiateli attentamente ed osservate gli esempi.

OBIETTIVI

Alla fine di questo capitolo, sarete in grado di:

- Dare la definizione dei termini: subroutine, SSI, MSI, LSI, allocare, stack, interrupt, interrogazione ciclica, software driver, interrupt vettorizzato, interrupt disabilitato, flag esterno, interrupt differito e registro di lettura.
- Spiegare come mascherereste una parola a 8 bit per ottenere lo stato del bit 5, o lo stato di uno qualunque degli altri sette bit.
- Spiegare come vengono caricate le informaziooni digitali e prelevate dallo stack del microcomputer 8080.
- Eseguire un calcolo approssimato che vi dirà come usare una subroutine.
- Descrivere l'interfacciamento di una tastiera ASCII.

DEFINIZIONI

Allocate (Allocare) — In un computer, assegnare le posizioni di memoria alle routine e alle subroutine principali, fissando così i valori assoluti degli indirizzi simbolici.

Breakpoint (Breakpoint) — Un punto di una routine specificato da un'istruzione, dal digit di un'istruzione, o da altre condizioni, in cui la routine può essere interrotta da un intervento esterno o da un monitor.

Breakpoint instruction (Istruzione di breakpoint) — Nella programmazione di un computer digitale, un'istruzione che, insieme ad un controllo manuale, fa sì che il computer si arresti.

Breakpoint switch (Switch di breakpoint) — Un interruttore azionato manualmente, che controlla la possibilità di realizzare dei breakpoint; viene usato principalmente nella messa a punto.

Diferred interrupt (Interrupt differito) — Un interrupt che si verifica in un certo momento, dopo che è stato settato un flag esterno.

Disable interrupt (Disabilitare l'interrupt) — Disabilitare il flag di interrupt all'interno del microprocessore.

Enable interrupt (Abilitare l'interrupt) — Abilitare il flag di interrupt all'interno del microprocessore.

External flag (Flag esterno) — Un circuito digitale, che contiene di solito un solo flip-flop, che indica una condizione esistente con riferimento ad un dispositivo di ingresso/uscita.

Flag — In un computer, l'indicazione che è stata portata a termine una particolare operazione. Inoltre, un flag è un flip-flop che può essere settato o azzerato in risposta ad operazioni che si stanno verificando all'interno di un microcomputer.

Immediate interrupt (Interrupt immediato) — Un interrupt che si verifica non appena viene settato un flag esterno.

Internal flag (Flag interno) — Un circuito digitale, contenente di solito un solo flip-flop, che indica una condizione che esiste all'interno di un microprocessore.

Interrupt (Interruzione) — In un computer, l'arresto della normale esecuzione di un programma in modo tale da permettere al programma stesso di essere ripreso nel punto

esatto in cui era stato interrotto. La sorgente dell'interrupt può essere interna ed esterna.

Interrupt flag (Flag di interrupt)	Un flip-flop interno al microprocessore che può essere abilitato o disabilitato da software, e che può rivelare una richiesta d'interruzione e memorizzare il verificarsi di un'interruzione. Nel seguito indicheremo con «interrupt» la parte attiva del fenomeno, ad esempio «il segnale di interrupt», e con «interruzione» la conseguenza.
Large-scale integration (Larga scala di integrazione)	Circuiti integrati digitali monolitici con una complessità tipica di 100 o più gate o circuiti equivalenti al gate. Il numero di gate usati per ogni chip per definire l'LSI dipende dal costruttore. Si abbrevia con LSI.[4]
Large-scale programs (Programmi a larga scala)	Programmi che contengono da 1000 a 10,000 byte di istruzione. Si abbrevia con LSP.
Mask (Maschera)	Una tecnica logica in cui determinati bit di una parola vengono cancellati o inibiti.
Medium-scale integration (Media scala di integrazione)	Circuiti integrati che funzionano come sistemi semplici, indipendenti, come i contatori a decade, le piccole memorie di lettura/scrittura, i decodificatori, i multiplexer, e i registri a scorrimento. Tali chip contengono di solito da 20 a 100 gate. Si abbrevia con MSI.
Medium-scale programs (Programmi a media scala)	Programmi che contengono da 100 a 1000 byte di istruzioni. Si abbrevia con MSP.
Multilevel interrupt (Interrupt a più livelli)	Un sistema di interruzione in cui esistono molte linee d'interruzione, ognuna delle quali è collegata ad un dispositivo di I/O separato. Il microcomputer non ha bisogno di scandire i dispositivi per determinare quale di essi ha causato la richiesta di interruzione.
Nesting (Nidificazione)	In un microcomputer, l'inclusione di una routine o di un blocco di dati dentro una routine o un blocco di dati più grande.
Polling (Interrogazione ciclica)	Interrogazione periodica di tutti i dispositivi che condividono una linea di comunicazione, per determinare quale ha bisogno di essere servito. Il multiplexer o la sezione di controllo invia una interrogazione che ha l'effetto di domandare al dispositivo selezionato, «Hai qualcosa da trasmettere»?

Priority (Priorità)	La condizione in cui vengono ordinati i dispositivi di I/O secondo la loro importanza, in modo che alcuni dispositivi abbiano la precedenza su altri.
Response time (Tempo di risposta)	Il tempo che intercorre fra la richiesta d'interruzione da parte di un dispositivo e il primo byte istruzioni del software driver che lo serve.
Sense (Esaminare)	Esaminare o determinare lo stato di alcuni componenti del sistema.
Service routine (Routine di servizio)	Nel software di un computer digitale, una routine progettata per assistere l'operazione in corso da parte del computer. Questo termine può significare anche una subroutine che serve un segnale d'interr. da un dispositivo esterno.
Single-line interrupt (Interrupt ad una sola linea)	Un sistema d'interruzione in cui vi è una sola linea di interrupt. Più dispositivi vanno collegati a questa linea con un'operazione di OR. Ogni ingresso al gate OR proviene da un dispositivo di I/O. Una volta ricevuto l'interrupt, il microcomputer deve scandire tutti i dispositivi per determinare quale di questi ha generato l'interrupt.
Small-scale integration (Piccola scala di integrazione)	Circuiti integrati che forniscono solo semplici gate, buffer, o flip-flop. Tali chip contengono di solito non più di dieci-venti gate. Si abbrevia con SSI.
Small-scale programs (Programmi a piccola scala)	Programmi che contengono fino a 100 byte di istruzioni. Si abbrevia con SSP.
Stack	Una regione di memoria che memorizza le informazioni temporanee, di solito i contenuti dei registri interni di un microprocessore.
Software driver	Una subroutine o una parte di un programma che trasferisce le informazioni fra il computer ed un dispositivo specifico di ingresso/uscita.
Status (Stato)	I contenuti dei registri interni, compresi i bit di flag, in un microprocessore durante l'esecuzione di un programma, in un dato momento.
Subroutine	Un piccolo sottoprogramma non facente parte della routine principale. Una subroutine di questo tipo viene inserita tramite un'operazione di salto nota come «call» (richiamo; si provvede a far ritornare li controllo al programma principale alla fine della subroutine).

Vectored interrupt (Interrupt vettorizzato)	Un sistema d'interruzione in cui l'interrupt provoca un salto diretto a quella parte del programma che serve l'interruzione stessa. Questo è il tipo di interrupt più veloce.
Vector bits (Bit di vettore)	I bit individuali che designano la posizione di branch in un'istruzione d'interrupt vettorizzato.
Very large-scale integration (Larghissima scala di integrazione)	Circuiti integrati digitali monolitici con una complessità tipica di 2.000 o più gate o circuiti equivalenti al gate. Abbreviato con VLSI.
Very large scale programs (Programma a larghissima scala)	Programmi che contengono più di 10.000 byte di istruzioni. Abbreviato con VLSP.

CHE COSA E' UNA SUBROUTINE?

Una subroutine si può così definire:

Subroutine	Un piccolo sottoprogramma non facente parte della routine principale. Tale subroutine viene inserita tramite un'operazione di salto nota come call; si provvede a che il controllo venga rimandato al programma principale alla fine della subroutine.

L'istruzione di call fa sì che il microcomputer trasferisca il controllo di programma alla subroutine selezionata. Le subroutine, generalmente, servono per eseguire specifici compiti ripetitivi; quando un compito è stato eseguito il microcomputer torna al programma principale da cui è partita la chiamata, e riprende l'operazione in quel punto. In qualunque momento venga usata un'istruzione di call, sia che essa sia condizionata o incondizionata, il microcomputer memorizza un indirizzo, di rientro nello stack, cosicché abbiamo modo di sapere a quale punto si deve ritornare quando l'esecuzione della subroutine è terminata. Lo stack è una regione di memoria di lettura/scrittura specializzata alla memorizzazione di informazioni temporanee, come gli indirizzi di rientro dalle subroutine ed i contenuti dei registri interni del microprocessore.

Quale indirizzo di rientro viene memorizzato nello stack? E' sempre lo stesso indirizzo a 16 bit? La Fig. 8-1 vi dà le risposte a queste due domande:

Una volta eseguita un'istruzione di call a tre byte e la subroutine associata, *vorremmo ritornare al byte di istruzione che segue immediatamente la call a tre byte.* Quindi, l'indirizzo a 16 bit del

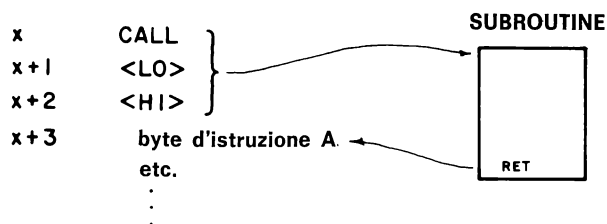



Fig. 8.1. L'indirizzo di ritorno che è memorizzato nello stack è x + 3, l'indirizzo del byte di istruzione che segue immediatamente l'istruzione call a 3 byte.

byte istruzione A, mostrato nella Fig. 8-1, è quello che viene memorizzato nello stack. Se «x» è l'indirizzo a 16 bit dell'istruzione di call, allora «x + 3» è l'indirizzo a 16 bit che viene memorizzato nello stack. Dato che si possono memorizzare solo otto bit alla volta nella memoria di lettura/scrittura di un microcomputer 8080, sono necessarie due posizioni consecutive nello stack per memorizzare tutto l'indirizzo di rientro a 16 bit. *Il microprocessore 8080 memorizza automaticamente questi due byte d'indirizzo di rientro nello stack in qualunque momento noi eseguiamo un'istruzione di call.* Il processo viene chiamato «pushing» (salvataggio) di un'indirizzo nello stack.

Tutte le subroutine devono finire, in un modo o nell'altro, con un'istruzione di rientro. Questa può essere condizionata o incondizionata. In entrambi i casi, un rientro fa sì che il microprocessore 8080 estragga l'indirizzo di rientro a 16 bit dallo stack. Questo indirizzo viene posto nel contatore di programma, come «x + 3» nel nostro esempio.

Non dobbiamo occuparci dei metodi che il chip 8080 usa per memorizzare o estrarre le informazioni dallo stack.

Comunque, dobbiamo essere sicuri di avere un'area dello stack a disposizione, nella memoria di lettura/scrittura, prima di tentare di richiamare qualunque subroutine. Possiamo posizionare lo stack in qualunque punto all'interno dei 65.536 byte di memoria disponibili, usando l'istruzione LXI SP, un'istruzione a tre byte in cui il secondo e il terzo byte contengono l'indirizzo dello stack a 16 bit, memorizzato in un registro a 16 bit all'interno del chip 8080, il *registro stack pointer.* Se volete, per esempio, settare l'indirizzo iniziale dello stack a HI = 003 e LO = 300, dovete usare questa sequenza di byte istruzioni:

Byte d'indirizzo di memoria LO	*Byte della istruzione ottale*	*Codice mnemonico*	*Commento*
000	061	LXI SP	Carica il registro stack pointer con i due byte d'indirizzo seguenti
001	300	300	Byte d'indirizzo LO
002	003	003	Byte d'indirizzo HI

Nell'esecuzione di questa istruzione, lo stack pointer conterrà l'indirizzo a 16 bit dato da HI = 003 e LO = 300. L'inizializzazione dello stack pointer è una delle operazioni iniziali più importanti che dovete eseguire, proprio all'inizio di un programma dell'8080.

Nel programma precedente, abbiamo supposto di avere solo 1K di memoria di lettura/scrittura. Abbiamo posizionato lo stack vicino alla «parte alta» di questa memoria, cioè vicino alle posizioni di memoria più alte. La ragione per cui lo abbiamo fatto è che lo stack, quando vengono inseriti dei dati, scende, come indirizzo di stack pointer, verso gli indirizzi di memoria inferiori. Infatti, l'indirizzo HI = 003 e LO = 300 è solo un indirizzo temporaneo dello stack. Se richiamassimo una sobroutine, osserveremmo che l'indirizzo di rientro (chiamato anche *indirizzo di linking*) è memorizzato come un byte di indirizzo ad 8 bit HI ad HI = 003 e LO = 277, ed un byte di indirizzo ad 8 bit LO ad HI = 003 e LO = 276. Lo stack pointer sarebbe allora decrementato di due. La maggior parte degli utenti dell'8080 non si accupano dell'ordine nel quale l'indirizzo viene inserito nello stack. A titolo informativo, il byte d'indirizzo (o di dati) HI viene sempre inserito per primo nello stack e il byte d'indirizzo (o di dati) LO per secondo. *Il primo byte che viene estratto dallo stack, è sempre il byte d'indirizzo (o di dati) LO.* Questo è in relazione al modo in cui il chip 8080 tratta le istruzioni a tre byte: il byte d'indirizzo (o di dati) LO viene caricato sempre per primo nel byte LO di un registro a 16 bit, come il contatore di programma, lo stack pointer, o la coppia di registri DE, BC, HL.

Lo stack pointer viene sempre decrementato (verso un indirizzo inferiore) *prima che le informazioni di dati o di indirizzi vengano inserite nello stack.* Lo stack pointer viene sempre incrementato (verso un indirizzo superiore) *dopo che le informazioni di dati o indirizzi vengono estratte dallo stack.* Se volessimo esaminare l'area dello stack dopo che è stata usata da un certo numero di subroutine, vi troveremmo ancora delle informazioni. E' importante questo? No davvero, dato che le informazioni appropriate sono state copiate molto tempo prima nel microprocessore 8080, dove erano necessarie. Le vecchie informazioni non hanno significato per il microprocessore; le nuove informazioni verranno scritte nelle precedenti in qualunque momento lo stack venga riadoperato.

UTILIZZO DELLO STACK PER LA MEMORIZZAZIONE DEI DATI E DELLO STATUS

La maggior parte delle subroutine impiega i registri universali o i flag interni al chip 8080. Si verificano dei problemi quando nei registri abbiamo dei dati che verranno usati anche dalle subroutine. Immaginate che il microcomputer sia nel mezzo di un calcolo numerico, quando un interrupt provoca il richiamo di una subroutine. Che cosa succede ai dati temporanei contenuti nei registri? vengono distrutti? Il calcolo deve essere rifatto? Possiamo

superare questa grave limitazione?

Chiaramente, dobbiamo avere la possibilità di memorizzare lo stato del programma, che si può definire come i contenuti dei registri interni, compresi i bit di flag, che si trovano all'interno di un mocroprocessore durante l'esecuzione di un programma in un dato momento. Per un chip 8080, lo stato è contenuto all'interno dei seguenti registri:

Contatore di programma
Accumulatore
Registro B
Registro C
Registro D
Registro E
Registro H
Registro L
Cinque bit di flag

Se voi siete a conoscenza di queste informazioni, l'esecuzione del programma può essere interrotta.

Memorizziamo lo stato del programma nello stack usando un gruppo di quattro istruzioni, chiamate istruzioni di push ed una sola istruzione di richiamo di subroutine. Alla fine di una subroutine, estraiamo lo stato del programma, usando quattro istruzioni di pop ed un'istruzione di rientro della subroutine. Le otto istruzioni di push e di pop sono le seguenti:

305	PUSH B	Memorizzare i contenuti della·coppia di registri B,C nello stack
325	PUSH D	Memorizzare i contenuti della coppia di registri D,E nello stack
345	PUSH H	Memorizzare i contenuti della coppia di registri H,L nello stack
365	PUSH PSW	Memorizzare la parola di stato del programma (PSW), cioè i contenuti dell'accumulatore e dei cinque flag, nello stack
301	POP B	Ripristinare i due byte presenti nella parte alta dello stack nella coppia di registri B,C
321	POP D	Ripristinare i due byte presenti nella parte alta dello stack nella coppia di registri D,E
341	POP H	Ripristinare i due byte presenti nella parte alta dello stack nella coppia di registri H,L
361	POP PSW	Ripristinare i due byte presenti nella parte alta dello stack nell'accumulatore e nei flip-flop dei flag

Tutte le operazioni di PUSH e di pop coinvolgono o il contatore di programma o una coppia di registri; esse sono operazioni a 16 bit che vengono eseguite su due byte a 8 bit, un byte di indirizzo

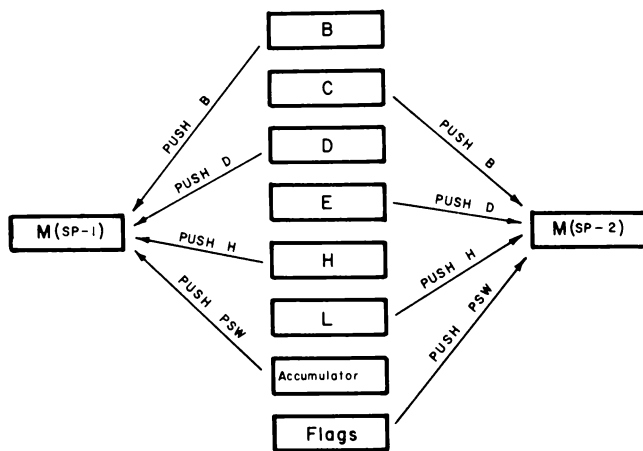
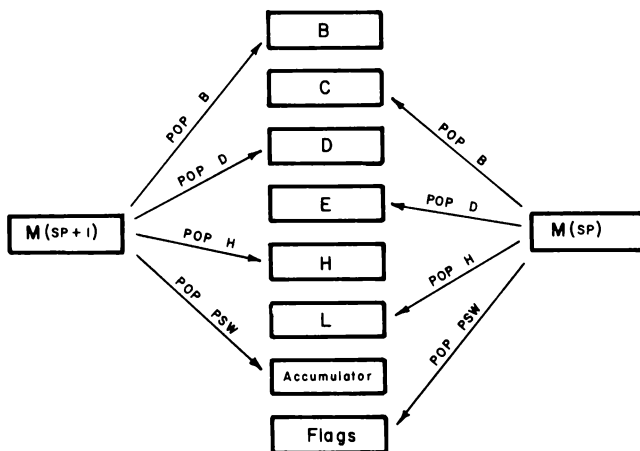



Fig. 8-2. Le operazioni di PUSH e di POP, ognuna delle quali coinvolge una coppia di registri e due locazioni di memoria.

o di dati LO e un byte d'indirizzo o di dati HI. La parola di stato del programma (PSW) è costituita dal registro accumulatore A, unitamente ai cinque flag del chip 8080 — carry, zero, parità, segno, e carry ausiliario — insieme a tre bit non usati. La Fig. 8.2 illustra le istruzioni di push e di pop. SP è l'indirizzo a 16 bit contenuto nel registro stack pointer all'interno del chip 8080.

Lo stack dell'8080 è uno stack last-in first-out (LIFO). Gli ultimi dati che vengono inseriti nello stack, sono i primi ad essere estratti. Lo potete verificare facilmente con l'aiuto della Fig. 8.3. Notate

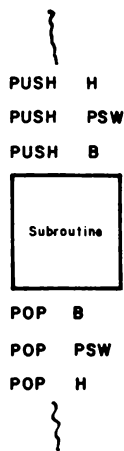



Fig. 8-3. Procedura esatta per inserire ed estrarre le informazioni nello e dallo stack.

che l'ordine di inserimento e di estrazione (push e pop) sono l'uno il contrario dell'altro. In questo esempio, la subroutine non ha influenzato i registri D o E, quindi non era necessario collocarli nello stack.

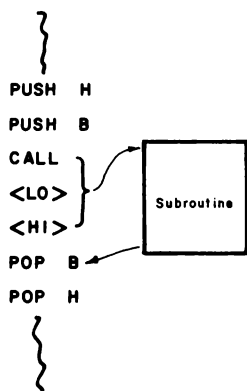
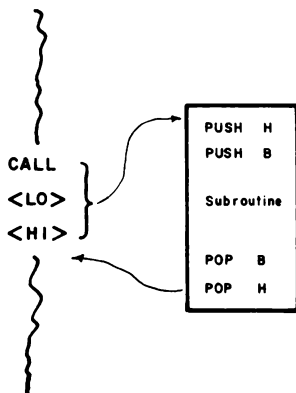
Le istruzioni di push e di pop permettono una grande flessibilità nella programmazione, dato che possiamo memorizzare i nostri dati temporanei prima che venga eseguita una subroutine ed estrarli dopo che la subroutine è stata eseguita. Questo solleva una domanda interessante: le istruzioni di push e di pop verrebbero posizionate prima o dopo l'istruzione di call, o verrebbero incluse nella subroutine? La Fig. 8-4 presenta le due alternative.

(A) In questo programma le istruzioni PUSH e POP sono presenti nel programma principale, come necessario.

(B) In questo programma le istruzioni PUSH e POP sono incorporate solo nella subroutine.

Fig. 8-4. Due alternative per posizionare le istruzioni PUSH e POP associate ad un richiamo di subroutine.

La risposta a questa domanda è la seguente: le includete nella subroutine. Ha più senso metterle nella subroutine piuttosto che ripeterle ogni volta che c'è un'istruzione di call. Lo stack memorizza così sia l'indirizzo di rientro (di *linking*) sia i dati presenti in tutte le coppie di registri coinvolte nella subroutine stessa. Quando assegnamo una posizione allo stack nella memoria di lettura/scrittura, dobbiamo accertarci che vi sia memoria sufficiente per le informazioni di stato associate con molti richiami di subroutine nidificate. Dato che lo stack aumenta scendendo verso gli indirizzi di memoria più bassi, di solito poniamo la posizione ini-

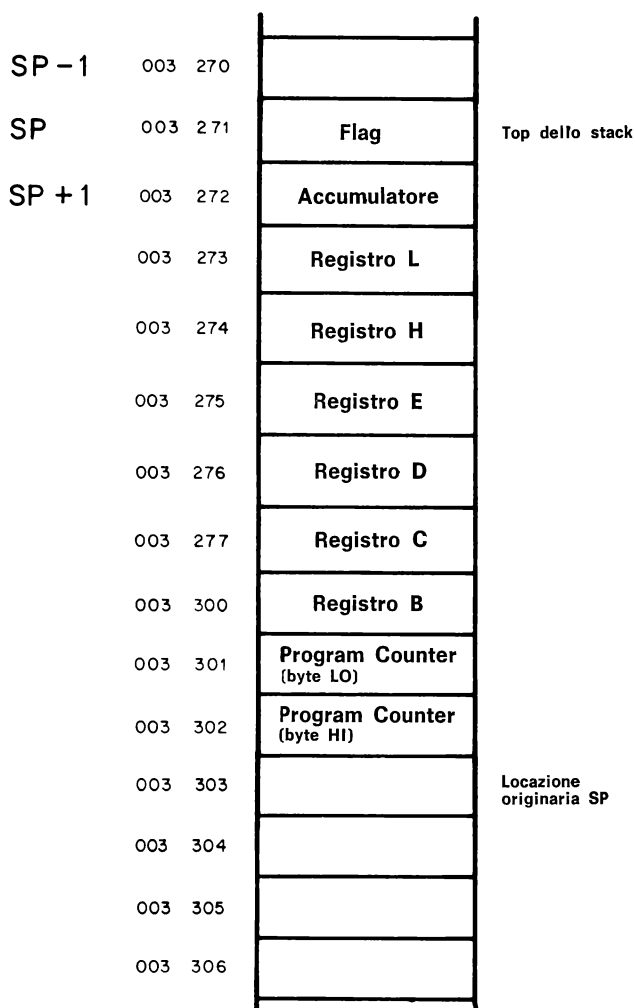



Fig. 8-5. Lo stack.

ziale dello stack all'indirizzo più alto disponibile della memoria di lettura/scrittura. Infine, le subroutine devono contenere lo stesso numero di istruzioni push e pop, in ordine inverso, nonché le istruzioni di rientro condizionato e incondizionato. Se non si verificano queste condizioni, il vostro stack potrebbe «scappar via» nella memoria di lettura/scrittura e distruggere qualunque programma o dato esistenti nella memoria stessa. Per un microcomputer 8080 che opera 2 MHz, occorrono solo pochi secondi per annullare 65.536 byte di memoria.

Concludiamo il nostro discorso con la Fig. 8-5, che mostra i contenuti dello stack dopo una call e quattro istruzioni di push successive. Notate che:

- Prima che richiamassimo la subroutine, la posizione originale dello *stack pointer* era HI = 003_8 e LO = 303_8.
- Dopo il caricamento del PSW nello stack, la nuova posizione dello stack pointer era HI = 003_8 e LO = 271_8. *Possiamo quindi concludere che lo stack pointer «punta» alla posizione di memoria utilizzata (riempita) nella parte alta dello stack stesso, non alla prima posizione libera (vuota).*
- Lo stack è del tipo *first-in last-out.* I primi byte inseriti erano i byte del contatore di programma successivi alla istruzione di call. Essi sono gli ultimi ad essere estratti dallo stack. Gli ultimi byte inseriti erano i contenuti dei flag e dell'accumulatore; essi sono i primi ad essere estratti.
- Il byte HI del contatore di programma o il byte HI di una coppia di registri è il primo ad essere inserito nello stack e l'ultimo dei due byte ad essere estratto. Quando viene estratta una coppia di registri, il byte LO della coppia viene estratto sempre per primo.
- In un'istruzione di push, il byte HI o il registro HI vengono memorizzati nella posizione dello stack SP —1 e il byte o il registro LO nella posizione SP —2. Lo stack pointer, SP, è decrementato di due, per produrre un nuovo valore dello stack pointer.
- In un'istruzione di pop, il byte o il registro LO vengono estratti dalla posizione SP dello stack e il byte o il registro HI viene estratto dalla posizione SP + 1. Lo stack pointer, SP, viene incrementato di due per produrre un nuovo valore dello stack pointer stesso.

QUANDO E' USATA UNA SUBROUTINE?

Quando usare una subroutine? Questo dipende da due fattori: il numero di passi di una subroutine, e il numero di volte che voi richiamate la subroutine dal programma principale. Ecco alcune regole generali:

- Se la subroutine proposta conterrà solo tre o quattro byte

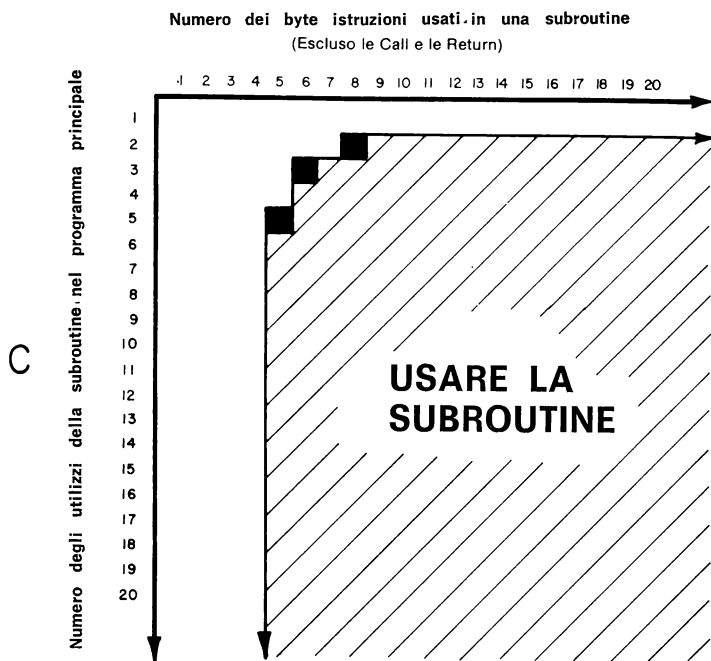



Fig. 8-6. Diagramma indicante quando si dovrebbe usare una subroutine.

istruzioni, non vi è un gran incentivo per voi a scrivere la subroutine, a prescindere da quante volte la chiamate nel programma principale.

- Se la subroutine proposta verrà usata solo una volta in un programma o in un gruppo di programmi, potreste incorporarla nel programma principale e non usarla come una subroutine.
- Se la subroutine proposta verrà usata da altri come componente dei loro programmi, dovreste scriverla in modo definito come subroutine e fare attenzione a dove viene posizionata in memoria.
- Se volete minimizzare il tempo che occorre per eseguire un gruppo di istruzioni, dovreste minimizzare il numero di subroutine presenti.
- Dovreste minimizzare la lunghezza del vostro programma principale e massimizzare la lunghezza delle vostre subroutine.

Fondamentalmente, il vostro obiettivo a lungo termine dovrebbe essere quello di acquisire un repertorio di subroutine che facilitino la programmazione del microcomputer.

Il fatto di decidere quando usare una subroutine può forse essere facilitato dall'uso della Fig. 8-6, in cui vi è *C*, il numero di volte in cui una subroutine viene usata in un programma principale, e X, il numero di byte istruzioni (meno le istruzioni di call e di rientro) usate nella subroutine. Abbiamo usato:

$$F = \frac{CX}{X + 4C}$$

per determinare quando si dovrebbe usare una subroutine. Quando $F > 1$, si dovrebbe usare una subroutine; quando $F < 1$, la subroutine non è necessaria. Si verifica una situazione imprecisa quando $F = 1$ (riquadri neri). Nell'equazione, la quantità CX è il numero totale di byte istruzioni usati in tutto il programma principale (meno la chiamata e le istruzioni di ritorno). La quantità $X + 4C$ è collegata al numero di byte istruzioni nell'intero programma principale richiesto per richiamare in modo ripetitivo la subroutine e per rientrare da quest'ultima. Il rapporto di questi due termini, F, è una misura della loro importanza relativa. Dovreste stabilire secondo il vostro giudizio quando usare una subroutine.

Gli autori pensano che le subroutine diventeranno sempre più importanti con il proliferare dei microcomputer. Nei paragrafi seguenti, vi presentiamo un punto di vista riguardante le subroutine.

INTEGRAZIONE HARDWARE: SSI, MSI, LSI e VSLI

Riassumiamo in breve l'incredibile progresso avvenuto nell'elettronica digitale negli ultimi dieci-quindici anni. Questo progresso si riflette principalmente negli acronimi SSI, MSI, LSI, VSLI, di cui daremo ora la definizione:

Small-scale integration (Piccola scala di integrazione [SSI])	Circuiti integrati che forniscono solo semplici gate o flip-flop. Tali chip contengono di solito non più di dieci-venti gate. Si abbrevia con SSI.
Medium-scale integration (Media scala di integrazione [MSI])	Circuiti integrati che funzionano come semplici, indipendenti, come i contatori a decade, le piccole memorie di lettura/scrittura, i decodificatori, i multiplexer, e i registri a scorrimento. Tali chip contengono di solito da 20 a 100 gate. Si abbrevia con MSI.
Large-scale integration (Larga scala di integrazione [LSI])	Circuiti integrati digitali monolitici con una complessità tipica di 100 o più gate o circuiti equivalenti al gate. Il numero di gate per ogni chip usato per definire LSI dipende dal costruttore. Si abbrevia con LSI.

Very large-scale integration (Larghissima scala di integrazione [VLSI]) Circuiti integrati digitali monolitici con una complessità tipica di 2000 o più gate o circuiti equivalenti ai gate. Si abbrevia con VLSI.

Le suddivisioni in scale sono alquanto arbitrarie. Ciononostante, il progresso nel campo dell'elettronica digitale si riflette moltissimo nella scala di integrazione raggiunta in un dato momento. A metà degli anni '60, l'integrazione su piccola scala emerse come una forza importante dell'elettronica. I chip erano costosi, e ne occorrevano molti per costruire uno strumento utile o un computer. Alla fine degli anni '60, l'integrazione su scala media incominciò a far diminuire i costi degli strumenti digitali riducendo il numero dei chip e il loro necessario. All'inizio degli anni '70 emerse l'integrazione su larga scala: grosse memorie e circuiti integrati UART. L'industria elettronica ebbe un po' di respiro quando apparve l'integrazione su larghissima scala, che dette l'avvio all'inevitabile impatto che l'elettronica avrà ancora in futuro. Le memorie ad accesso casuale a 4K e i microprocessori a 8 bit danno l'opportunità di costruire computer potenti per solo poche centinaia di dollari. Prossimi a venire sono i computer costruiti su di un solo chip, con 4K di RAM e capacità di moltiplicazione/divisione veloci. Ci arriveremo solo tra pochi anni.

INTEGRAZIONE SOFTWARE: SSP, MSP, LSP e VLSP

Potete osservare i progressi nel campo dell'elettronica digitale anche vedendo quello che succede ai chip di memoria. Questi sono gli orientamenti:

- I nuovi chip di memoria conterranno più memoria per ogni singolo chip. Le RAM a 4K e le ROM a 16K erano comuni nel 1975. Non molto più tardi arrivano le RAM a 16K e le ROM a 64K. Avremo tra poco le RAM a 64K e le ROM a 256K. Dovreste notare che occorre solo una parola d'indirizzo di memoria a 18 bit per indirizzare qualunque posizione fra le 262.144 posizione diverse.
- I chip di memoria veloci diventeranno più veloci. Esiste un tradeoff fra la lunghezza di memoria e la velocità di base di una cella di memoria, ma i miglioramenti nella tecnologia stanno aprendo la strada alle grosse memorie che sono più veloci dei chip odierni.

La velocità e la lunghezza di memoria sono le variabili importanti per il futuro. La velocità della memoria determina il tempo di esecuzione di un'istruzione in un computer. Più veloce è la memoria, maggiore è il numero di istruzioni che possono venire eseguite in

un dato periodo, e più potenti diventano i programmi che si possono usare. Gli autori pensano che un buon metro di misura del progresso che deve ancora avvenire per i microcomputer sia la lunghezza dei programmi usati. Insieme alle abbreviazioni, vi suggeriamo le seguenti indicazioni della complessità dei programmi:

Small-scale programs (Programmi a piccola scala [SSP])	Programmi che contengono fino a 100 byte di istruzioni. Si abbrevia con SSP.
Medium-scale programs (Programmi a scala media [MSP])	Programmi che contengono da 100 a 1000 byte di istruzioni. Si abbrevia con MSP.
Large scale programs (Programmi a larga scala [LSP])	Programmi che contengono da 1000 a 10.000 byte di istruzioni. Si abbrevia con LSP.
Very large-scale programs (Programmi a larghissima scala [VLSP])	Programmi che contengono più di 1.000 byte di istruzioni. Si abbrevia con VLSP.

Per eseguire un programma di 10.000 byte istruzioni, il vostro computer deve avere sia una memoria molto grossa che una memoria molto veloce. Con un computer a 8 bit, 10.000 byte di istruzione corrispondono a 80.000 bit di memoria, o 80 kilobit di memoria.

Avete tentato di scrivere un programma di 1.000 byte di istruzioni? Vi occupa decisamente un certo tempo, specialmente se il programma è scritto attentamente ed esegue delle funzioni interessanti.

Quello a cui si vuole arrivare con questa discussione è sottolineare l'importanza delle subroutine. Una subroutine può essere di qualunque lunghezza. Se c'è un numero di subroutine diverse facilmente disponibile, vi faranno risparmiare tempo dato che non dovete riscriverle. Quindi, un programma di 1000 byte di istruzioni potrebbe essere molto facile da scrivere, ammesso che consista di 200 byte di programma principale e di 800 byte di subroutine che altri hanno preparato per voi.

Dovreste pensare alle subroutine con ogni programma MSP che scrivete. Quali funzioni di programma sono necessarie? Ritardi di tempo? Addizioni, sottrazioni, moltiplicazioni, o divisioni a più bit? Metteteli in subroutine. Avrete bisogno di acquisire i dati da uno strumento a quattro cifre bcd? Scrivete una subroutine di ac-

quisizione dati in cui potete settare nel programma principale il numero di canali desiderati e il tempo fra i singoli canali. Avete bisogno di mettere in uscita le informazioni si di una telescrivente per mezzo di un chip UART e di un current loop di 20 mA? Scrivete un *software driver,* una subroutine che gestisce le richieste di trasferimento dati.

LE ISTRUZIONI DI SUBROUTINE DELL'8080

Vi sono 26 istruzioni di subroutine nel set di istruzioni del microcomputer 8080:

- Un'istruzione di richiamo incondizionato a tre byte, CALL

315	CALL <B2> <B3>	Richiamo incondizionato della subroutine alla posizione di memoria indirizzata dai byte B2 e B3

- Un'istruzione di rientro incondizionato ad un byte, RET

311	RET	Rientro incondizionato dalla subroutine

- Otto diverse istruzioni di ripristino, ad un byte, RST

307	RST 0	Richiama la subroutine a HI = 000_8 e LO = 000_8
317	RST 1	Richiama la subroutine a HI = 000_8 e LO = 010_8
327	RST 2	Richiama la subroutine a HI = 000_8 e LO = 020_8
337	RST 3	Richiama la subroutine a HI = 000_8 e LO = 030_8
347	RST 4	Richiama la subroutine a HI = 000_8 e LO = 040_8
357	RST 5	Richiama la subroutine a HI = 000_8 e LO = 050_8
367	RST 6	Richiama la subroutine a HI = 000_8 e LO = 060_8
377	RST 7	Richiama la subroutine a HI = 000_8 e LO = 070_8

- Otto diverse istruzioni di rientro condizionato ad un byte

300	RNZ	Rientra dalla subroutine se il flag di zero è a livello logico 0
310	RZ	Rientra dalla subroutine se il flag di zero è a livello logico 1
320	RNC	Rientra dalla subroutine se il flag di carry è a livello logico 0

330 RC		Rientra dalla subroutine se il flag di carry è a livello logico 1
340 RPO		Rientra dalla subroutine se il flag di parità è a livello logico 0
350 RPE		Rientra dalla subroutine se il flag di parità è a livello logico 1
360 RP		Rientra dalla subroutine se il flag di segno è a livello logico 0
370 RM		Rientra dalla subroutine se il flag di segno è a livello logico 1

- Otto diverse istruzioni di richiamo condizionato a tre byte

304 CNZ	<B2> <B3>	Richiama la subroutine alla posizione di memoria indirizzata dai byte B2 e B3, se il flag di zero è a livello logico 0
314 CZ	<B2> <B3>	Richiama la subroutine alla posizione di memoria indirizzata dai byte B2 e B3, se il flag di zero è a livello logico 1
324 CNC	<B2> <B3>	Richiama la subroutine alla posizione di memoria indirizzata dai byte B2 e B3, se il flag di carry è a livello logico 0
334 CC	<B2> <B3>	Richiama la subroutine alla posizione di memoria indirizzata dai byte B2 e B3, se il flag di carry è a livello logico 1
344 CPO	<B2> <B3>	Richiama la subroutine alla posizione di memoria indirizzata dai byte B2 e B3, se il flag di parità e a livello logico 0
354 CPE	<B2> <B3>	Richiama la subroutine alla posizione di memoria indirizzata dai byte B2 e B3, se il flag di parità e a livello logico 1
364 CP	<B2> <B3>	Richiama la subroutine alla posizione di memoria indirizzata dai byte B2 e B3, se il flag di segno è a livello logico 0
374 CM	<B2> <B3>	Richiama la subroutine alla posizione di memoria indirizzata dai byte B2 e B3, se il flag di segno è a livello logico 1

Potete far riferimento al Capitolo 3 o all'appendice, dove le istruzioni suddette sono discusse in modo più dettagliato.

LE ISTRUZIONI DI STACK DELL'8080

Esistono quattordici operazioni relative allo stack, che si aggiungono alle istruzioni di subroutine date nel paragrafo precedente. Viene considerata istruzione relativa allo stack quella in cui la posizione dello stack pointer o i contenuti dello stack, o entrambi, vengono alterati. Invece di parlarvi dettagliatamente delle operazioni dello stack, ne faremo semplicemente un elenco e potrete fare riferimento al Capitolo 3 o all'appendice per ulteriori particolari.

- Un'istruzione di caricamento immediato a tre byte, LXI SP

061 LXI <B2> <B3>	Carica in modo immediato i due byte B2 e B3 nel registro stack pointer

- Un'istruzione di trasferimento dei dati, SPHL

371 SPHL	Trasferisci i contenuti dei registri H e L nel registro stack pointer

- Due istruzioni di incremento e decremento, INX SP e DCX SP

063 INX SP	Incrementa i contenuti del registro stack pointer di uno
073 DCX SP	Decrementa i contenuti del registro stack pointer di uno

- Un'istruzione aritmetica, DAD SP

071 DAD SP	Somma i contenuti del registro stack pointer ai contenuti della coppia di registri H e L e memorizzali nella coppia di registri H e L

- Un'istruzione di scambio dei contenuti dello stack, XTHL

343 XTHL	Scambia la parte alta dello stack con i contenuti della coppia di registri H e L

- Quattro istruzioni di estrazione dello stack ad un solo byte, POP

301 POP B	Estrai lo stack e trasferisci i contenuti nella coppia di registri B e C. Incrementa il registro stack pointer di due
321 POP D	Estrai lo stack e trasferisci i contenuti nella coppia di registri D e E. Incrementa il registro stack pointer di due
341 POP H	Estrai lo stack e trasferisci i contenuti nella coppia di registri H e L. Incrementa il registro stack pointer di due

361 POP PSW	Estrai lo stack e trasferisci i contenuti nell'accumulatore e rimemorizza i flag di condizione. Incrementa il registro stack pointer di due

- Quattro istruzioni di inserimento nello stack, ad un solo byte

305 PUSH B	Inserisci i contenuti della coppia di registri B e C nello stack. Decrementa il registro stack pointer di due
325 PUSH D	Inserisci i contenuti della coppia di registri D e E nello stack. Decrementa il registro stack pointer di due
345 PUSH H	Inserisci i contenuti della coppia di registri H e L nello stack. Decrementa il registro stack pointer di due
365 PUSH PSW	Inserisci i contenuti dell'accumulatore e dei flag di condizione nello stack. Decrementa il registro stack pointer di due

ALLOCAZIONE DI MEMORIA

Siamo quasi pronti per rispondere nei minimi particolari alla domanda: «Che cos'è uno stack?». Prima di farlo, comunque, dovremo prendere brevemente in considerazione *l'allocazione* della memoria in un microcomputer. Il termine *allocare* significa:

Allocare (Allocate)	In un computer, assegnare le posizioni di memoria alle routine e subroutine principali, fissando così i valori assoluti degli indirizzi simbolici.

Fondamentalmente, allocazione è il modo in cui suddividiamo una sezione o un blocco di memoria in gruppi o locazioni minori, ognuno dei quali contiene una delle cose elencate:

- *Un programma principale,* al quale il microcomputer inizia l'esecuzione. Non è necessario che il programma principale sia lungo. Infatti, può consistere principalmente di richiami delle subroutine e di loop di attesa.
- Una o più *subroutine,* dove viene eseguito la maggior parte del lavoro di elaborazione. Le subroutine possono essere di delay, routine di calcolo aritmetico, software driver, o routine di servizio, per nominare solo alcuni tipi.
- Fino a otto piccole *routine di servizio interrupt,* che rispondono a diverse condizioni d'interrupt e istruzioni di ripristino,

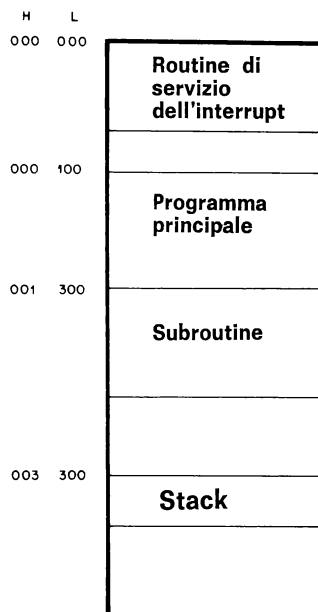



Fig. 8-7. Uno schema ipotetico di allocazione di memoria per la prima sezione di memoria a 1K. Qui lo stack è posizionato nella parte alta della sezione di memoria, alquanto staccato da tutte le altre regioni di memoria.

RST, che si trovano tutte insieme nel computer.

- Un solo *stack*, dove vengono memorizzate le informazioni temporanee mentre vengono eseguite le subroutine. Potete considerare lo stack come il contabile del computer. Esso prende nota di tutte le subroutine che sono state richiamate.

Nella Fig. 8-7 sono illustrate molte posizioni ipotetiche per quanto riguarda i gruppi suddetti, e sono tutte sistemate nella prima sezione di memoria a 1K del microcomputer. Notate che non è necessario affollare tutti insieme i diversi tipi di zone di memoria. Lo stack, se possibile, dovrebbe essere separato dalle altre regioni di programma. Questo perché esso va aumentando dalla fine della memoria verso l'inizio.

METODI OPERATIVI DEL MICROCOMPUTER

In questo libro, abbiamo presentato solo semplici programmi, ed abbiamo costruito interfacce molto semplici per il microcomputer, usando solo pochi circuiti integrati della serie 7400 (più il buffer/latch 8212 e il buffer 8095). Anche se in questo capitolo procederete nello stesso modo, il mondo reale dell'interfacciamento del microcomputer esigerà da voi almeno un aumento, in ordine di grandezza, della sofisticazione dei programmi e delle in-

terfacce. Nella realtà, vi sarà richiesto di interfacciare strumenti e macchine e di creare una comunicazione fra il microcomputer e una telescrivente, un display CRT, un altro microcomputer, un minicomputer, un'unità nastro a cassetta, un disco magnetico, e simili. In breve, si può richiedere che un solo microcomputer 8080 comunichi con un certo numero di dispositivi di I/O, che occupano tutti l'attenzione del microcomputer.

Come fa un solo microcomputer a trattare molti dispositivi di I/O? *Si occupa di un solo dispositivo alla volta!* Comunque, il modo in cui si occupa di «un solo dispositivo» può essere molto interessante. Vi sono due modi principali di eseguire le operazioni per il microcomputer: *l'interrogazione ciclica e l'interrupt.* Parleremo ora di tutti e due.

Operazione a Interrogazione Ciclica

Graf ha definito *l'interrogazione ciclica* in questo modo:

Polling (Interrogazione ciclica) — Interrogazione periodica di ognuno dei dispositivi che condividono una linea di comunicazioni, per determinare quale ha bisogno di essere servito. Il multiplexer o la sezione di controllo inviano una «interrogazione» che ha l'effetto di domandare al dispositivo selezionato: *«Hai qualcosa da trasmettere?».*
Quando un microcomputer *serve* un dispositivo, non fa altro che scambiare le informazioni digitali con il dispositivo in un modo che è prescritto da alcuni segmenti del programma. Il segmento particolare del programma viene spesso chiamato *software driver,* che possiamo definire in questo modo:

Software driver — Una subroutine o una parte di un programma, che trasferisce le informazioni fra il computer ed un dispositivo specifico di ingresso/uscita.

Nell'operazione di interrogazione ciclica il microcomputer controlla semplicemente in sequenza i dispositivi legati al microcomputer stesso, cercando i singoli dispositivi che hanno bisogno di essere serviti. Quando trova uno di questi dispositivi, interrompe la sequenza, chiama un software driver ed attua il servizio. Quando ha finito, prosegue con la sequenza attraverso gli altri dispositivi.

L'operazione di interrogazione ciclica è più che mai utile con i dispositivi relativamente lenti che non hanno bisogno di essere serviti spesso, o per lo meno possono aspettare di essere serviti. Ci si avvantaggia dalle differenze di velocità fra il microcomputer e i dispositivi interrogati ciclicamente, ognuno dei quali po-

trebbe essere servito in una sola frazione di secondo. In 100 ms, il microcomputer 8080 può eseguire circa 20.000 istruzioni.

Operazione d'Interruzione

Graf ha definito l'interruzione in questo modo:

Interrupt (Interruzione) In un computer, l'arresto della normale esecuzione di un programma in modo tale da permettere al programma stesso di essere ripreso nel punto esatto in cui era stato interrotto. La sorgente dell'interrupt può essere interna o esterna.

L'operazione di interrupt è molto più sofisticata e può ovviare alla maggior parte degli svantaggi inerenti l'operazione di interrogazione ciclica. Nell'operazione d'interrogazione ciclica:

- Il microcomputer trascorre molto del suo tempo a prendere in esame in modo sequenziale i dispositivi ad esso collegati.
- Una volta che un dispositivo è stato servito, esso deve aspettare il suo turno fino a che tutti gli altri dispositivi sono stati passati in rassegna uno dopo l'altro e, se necessario, anche serviti. In altre parole, non esiste priorità in questa operazione. Tutti i dispositivi sono trattati nello stesso modo.
- Il *tempo di risposta* del momento in cui un dispositivo vuole essere servito e quello in cui è effettivamente servito, può essere sostanzialmente, per lo meno nei microcomputer standard.

Al contrario, nell'operazione interrupt:

- Il microcomputer, può passare molto del suo tempo in un loop di attesa, aspettando che un dispositivo chieda di essere servito.
- Ci può essere *priorità* nell'operazione d'interrupt. I dispositivi più importanti possono essere serviti più spesso di quelli meno importanti.
- Il *tempo di risposta* che intercorre dal momento in cui un dispositivo importante vuol essere servito a quello in cui viene servito, può essere molto breve, anche nei microcomputer standard. Con il microcomputer 8080, questo tempo di solito non supera i 10 µs.
- Il software diventa molto più complesso.

I termini *priorità* e *tempo di risposta* si possono così definire:

Priority (Priorità) La condizione in cui i dispositivi di ingresso/uscita vengono disposti in ordine d'importanza in modo che alcuni dispositivi abbiano la precedenza su altri.

Response time (Tempo di risposta) Il tempo che intercorre fra la richiesta d'interruzione da parte del dispositivo e il primo byte istruzioni del software driver che lo serve.

L'operazione di interrupt è raffigurata schematicamente nella Fig. 8-8. Nei microcomputer 8080, un dispositivo di I/O che interrompe il microcomputer invia sia un impulso di interrupt — un impulso solo — che un byte istruzioni a 8 bit che dice al microcomputer che cosa deve fare dopo che è stato interrotto. Con il microcomputer 8080 questa istruzione a 8 bit è una delle otto diverse istruzioni di ripristino (RST), che sono istruzioni ad un solo byte che richiamano una subroutine ad una delle otto diverse posizioni di memoria: HI = 000_8 e LO = 000, 010, 020, 030, 040, 050, 060, o 070_8.

Il microprocessore 8080 permette istruzioni d'interrupt a più byte. Non parleremo di questa possibilità in questo libro, ma tenete presente che esiste.

Chiaramente, nell'operazione d'interrupt, il microcomputer 8080 serve i suoi dispositivi su richiesta, con l'aiuto delle considerazioni di priorità. Se nessun dispositivo ha bisogno di essere servito, il microcomputer passa semplicemente il tempo in un loop di attesa o esegue altri compiti legati al suo software. Deve essere chiaro che le *interruzioni sono complesse, richiedono software addizionali, e sono difficili da mettere a punto.* Vi sono molte persone che riducono l'uso delle interruzioni al minimo.

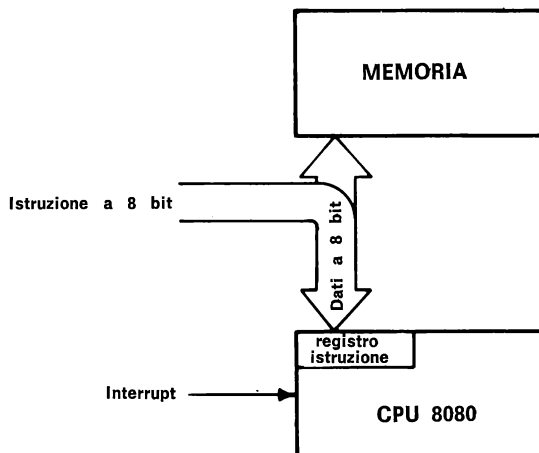



Fig. 8-8. Durante un'interrupt un'istruzione a 8 bit viene immessa nel registro di istruzioni nell'8080.

PRINCIPALI TIPI DI INTERRUPT

Esistono tre tipi di interrupt: *ad una sola linea, a più livelli, e vettorizzato.*

Single-line interrupt (Interrupt ad una sola linea) Un sistema d'interruzione in cui c'è una sola linea d'interrupt. Più dispositivi vanno collegati a questa linea con un'operazione di OR. Ogni ingresso al gate OR proviene da un dispositivo di I/O. Una volta ricevuto l'interrupt, il microcomputer deve scandire tutti i dispositivi per determinare quale di essi ha generato l'interrupt.

Multilevel interrupt (Interrupt a più livelli) Un sistema d'interruzione in cui esistono molte linee d'interruzione, ognuna delle quali è collegata ad un dispositivo di I/O separato. Il microcomputer non ha bisogno di scandire i dispositivi per determinare quale di essi ha causato la richiesta di interruzione.

Vectored interrupt (Interrupt vettorizzato) Un sistema d'interruzione in cui l'interrupt provoca un salto diretto a quella parte del programma che serve l'interruzione stessa. Questo è il tipo di interrupt più veloce.

Questi tre tipi di interrupt sono mostrati schematicamente nella Fig. 8-9. L'interrupt ad una sola linea è un tipo comune per i microcomputer, e facile da implementare. Non ci sono limiti al numero di dispositivi su cui si può eseguire un'operazione di OR verso una sola linea d'interrupt. Nella Fig. 8-9A sono mostrati tre dispositivi. Comunque, più sono i dispositivi, più tempo occorre per scandirli. La priorità è determinata dall'ordine in cui i dispositivi vengono scanditi sotto il controllo del software.

L'interrupt a più livelli, che vedete nella Fig. 8-9A, è una tecnica che ha effetto se vi è un numero sufficiente di pin d'interrupt sul microprocessore. Questo accade raramente. I microprocessori esistenti non hanno più di quattro pin d'interrupt.

La tecnica dell'interrupt vettorizzato (Fig. 8-9C) permette di saltare direttamente a quella parte del programma, quasi sempre una subroutine chiamata *subroutine di servizio,* che serve immediatamente l'interrupt. L'impulso d'interrupt viene prima inviato al microprocessore, seguito da un'istruzione ad un solo byte di 8 bit che viene «*forzata*» nel microprocessore immediatamente dopo l'impulso d'interrupt. L'Intel 8080 fornisce otto diverse istruzioni di ripristino, come abbiamo detto in un precedente paragrafo:

307	RST 0	Richiama la subroutine a HI = 000_8 e LO = 000_8
317	RST 1	Richiama la subroutine a HI = 000_8 e LO = 010_8
327	RST 2	Richiama la subroutine a HI = 000_8 e LO = 020_8
337	RST 3	Richiama la subroutine a HI = 000_8 e LO = 030_8
347	RST 4	Richiama la subroutine a HI = 000_8 e LO = 040_8
357	RST 5	Richiama la subroutine a HI = 000_8 e LO = 050_8
367	RST 6	Richiama la subroutine a HI = 000_8 e LO = 060_8
377	RST 7	Richiama la subroutine a HI = 000_8 e LO = 070_8

Fig. 8-9. Tre diversi tipi di tecniche di interrupt.

Dovreste osservare che solo tre degli otto bit sono cambiati nelle suddette istruzioni di ripristino (RST). L'Intel Corporation, avvantaggiandosi di questo fatto, ha fornito un chip speciale, l'8214 priority interrupt control unit, che contiene tutta la logica necessaria per:

- Causare un'interrupt al microprocessore 8080.
- Forzare tre bit di vettore, in combinazione con altri cinque, nel microprocessore 8080, immediatamente dopo un'interruzione.
- Fornire otto livelli di priorità, es. le otto istruzioni RST.
- Fornire le uscite a collettore aperto sia per le uscite d'inter-

rupt che per quelle di vettore, permettendo così una capacità d'interrupt per più di otto dispositivi.

Le specifiche del costruttore per il chip sono mostrate nel Capitolo 1. Ci si potrebbe aspettare che l'hardware che serve per l'interrupt all'interno del microprocessore migliori col tempo. I chip 8080A e 8228 permettono che istruzioni a più byte vengano inserite immediatamente dopo un'interruzione. Comunque, sono necessari degli speciali circuiti digitali al chip per avvantaggiarsi di questa caratteristica. Una nuova interfaccia, l'8259 Programmable interrupt controller, esegue alcune di queste funzioni.

ISTRUZIONI DI ABILITAZIONE E DISABILITAZIONE INTERRUPT

Non è raro per un computer avere molti dispositivi di I/O collegati a sè, ognuno dei quali ne richiede costantemente l'attenzione. L'uso di livelli di priorità aiuta a risolvere il problema di servire questi dispositivi, ma talvolta può capitare che il computer voglia sapere che è arrivato l'interrupt, senza che sia necessario servirlo. Questo accade specialmente mentre è in atto una funzione di software complessa e legata al tempo. A questo scopo, vi è una parte dei circuiti interni al microprocessore e del suo software che permette al computer stesso di porgere un «orecchio sordo» a tutte le interruzioni. Si tratta di *un flip-flop o flag d'interrupt* (i termini hanno lo stesso significato) che può essere abilitato o disabilitato da istruzioni ad un solo byte. Ecco alcune definizioni:

Interrupt flag, interrupt flip-flop (Flag/flip-flop d'interrupt) — Un flip-flop interno al microprocessore che può essere abilitato o disabilitato dal software e che può rivelare una richiesta di interruzione e memorizzare il verificarsi di un'istruzione.

Disable interrupt (Disabilitare l'interrupt) — Disabilitare il flag d'interrupt all'interno del microprocessore.

Enable interrupt (Abilitare l'interrupt) — Abilitare il flag d'interrupt all'interno del microprocessore.

Le istruzioni di abilitazione e disabilitazione dell'interrupt nel set di istruzioni dell'8080 sono:

373 EI — Abilita il sistema d'interruzione interno al microprocessore, dopo *l'esecuzione dell'istruzione successiva.*

363 DI — Disabilita il sistema d'interruzione interno al microprocessore, dopo *l'esecuzione dell'istruzione successiva.*

Se il flag d'interrupt è abilitato, un'interruzione verrà servita immediatamente. Comunque, anche se il flag d'interrupt è disabilitato, il

computer può ancora rivelare se è avvenuta o meno un'interruzione. Se è avvenuta un'interruzione mentre il flag di interrupt era disabilitato, dovrebbe verificarsi non appena il flag viene abilitato. Viene ricordato solo un evento d'interruzione.

FLAG ESTERNI

Il termine *«flag»* è stato definito nel Capitolo 1 come:

Flag — In un computer, un'indicazione che è stata completata una particolare operazione.[4] Inoltre, un flip-flop che può essere settato o azzerato in risposta ad operazioni che vengono eseguite nel microcomputer.

Tale flag si può chiamare più correttamente *flag interno,* dato che è situato all'interno dei circuiti del microprocessore 8080. Un circuito d'interfaccia estremamente importante, ma molto semplice, è un *flag esterno,* che si può così definire:

External flag (Flag esterno) — Un circuito digitale, contenente di solito un solo flip-flop, che indica una condizione esistente con riferimento ad un dispositivo di ingresso/uscita.

Il «flag interno» si può definire in modo analogo:

Internal flag (Flag interno) — Un circuito digitale, contenente di solito un solo flip-flop, che indica una condizione esistente internamente al microprocessore.

La parola veramente importante delle precedenti definizioni è *«condizione».* Un flag è in grado di rivelare un cambiamento di condizione quasi istantaneamente; è poi in grado di ricordare questa condizione fino a che essa viene azzerata da un impulso di clock proveniente dal microprocessore o, se il flag è interno, fino a che il programma non cambia lo stato logico del flag. Potrebbe essere più appropriato chiamare un flag esterno *«flag di condizione esterno».* Questo termine, comunque, è ridondante.

I flag esterni indicano le condizioni esistenti con riferimento ad un dispositivo di ingresso/uscita; quindi, essi sono in grado di sincronizzare le operazioni del dispositivo con quella del microcomputer. In un microcomputer con un'interfaccia che funzioni correttamente, essi non sono meno importanti degli impulsi di selezione dispositivo, di cui abbiamo parlato in precedenza. Ecco vari (ma non tutti) tipi di condizioni che il flag è in grado di indicare:

- I dati sono a disposizione per essere inseriti nel microcomputer dal dispositivo esterno.

- Il dispositivo è pronto per accettare i dati in uscita dal microcomputer.
- Il dispositivo ha terminato un'operazione, ed è pronto per iniziarne una nuova.
- Il dispositivo è occupato e non vuole essere disturbato.
- C'è stato un abbassamento di corrente; il microcomputer deve reagire immediatamente per conservare i dati e i programmi utili.
- Il dispositivo non sta operando, per il momento.
- Una quantità controllata ha superato la zona pericolosa.
- Il valore di una quantità controllata è troppo alto.
- Il valore di una quantità controllata è troppo basso.

Nella Fig. 8-10 vedete uno schema di un tipico circuito di flag esterno. Gli autori preferiscono questo circuito a quello che può essere costruito da un flip-flop J-K 7476. Il flip-flop 7474 è positive edge triggered, mentre il flip-flop 7476 è level triggered, ma entrambi i flip-flop sono efficaci.

Un fronte positivo temporizza l'uscita Q del flip-flop, a livello logico 1. Questo stato logico viene inserito in un gate NAND a due ingressi 7400, che inverte i segnali ed applica un livello logico 0 all'ingresso di interrupt dell'8080. *Mentre il chip 8080 richiede un livello logico 1 per generare un'interruzione al pin 14 sul chip, la figura mostra un livello logico 0 che genera l'interruzione. Ciò significa che l'ingresso d'interrupt viene invertito prima di entrare nell'8080. La condizione bassa attiva (livello logico 0) è la condizione progettata dagli autori, nel microcomputer 8080 che hanno usato.*

Il secondo ingresso del gate NAND 7400 si può usare per effettuare un gate del segnale di interrupt nel microcomputer. Se il flag all'interno del microcomputer è stato precedentemente abili-

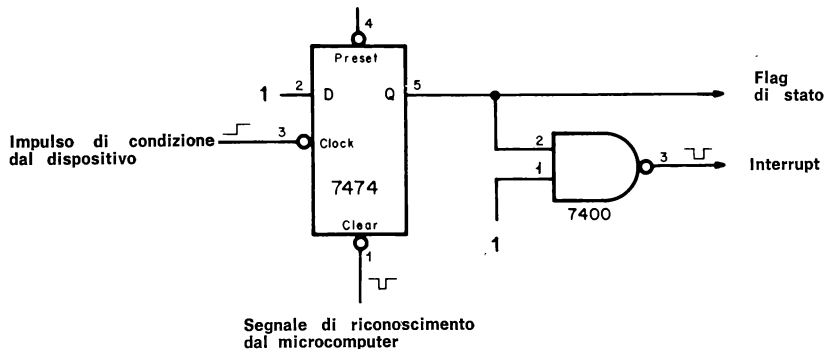



Fig. 8-10. Il circuito di un flag esterno.

tato da un'istruzione EI, il microcomputer viene interrotto ed il programma salta ad una subroutine, che è specificata da un'istruzione a 8 bit che viene «forzata» nel computer tramite circuiti esterni. Tale istruzione è di solito un'istruzione di ripristino, RST n, *ma può essere qualunque altra istruzione ad un solo byte.* Dovreste tener presente il fatto che il nuovo microprocessore 8080A, messo a disposizione dalla Intel e da altre case costruttrici, permette di forzare un'istruzione a più byte, come un richiamo, durante un'interruzione. E' comunque necessario dell'hardware in più, compreso il chip 8228, per farlo. Non è consigliabile portare lunghe linee o fili direttamente dalle uscite dei flip-flop e dei latch. E' stato aggiunto il gate NAND a due ingressi 7400, per fornire la capacità di pilotaggio e per creare un buffer all'uscita del flip-flop.

Nella Fig. 8-11 vedete una rappresentazione più utile del circuito di un flag esterno. Questo circuito si chiama «flag esterno 0» e vengono identificati i seguenti ingressi e uscite del flag:

- L'impulso di condizione, che proviene di solito da uno strumento o da un dispositivo.
- L'ingresso di azzeramento, $\overline{\text{CLR-0}}$. Un impulso di clock negativo azzererà il flag.
- L'ingresso di preset, $\overline{\text{PR-0}}$. Un impulso di clock negativo setterà il flag.
- L'uscita dei flag, STAT-0. Questa uscita viene inviata solitamente come ingresso al microcomputer attraverso una porta d'ingresso three-state.
- L'uscita d'interrupt del flag, $\overline{\text{INT-0}}$. Un impulso di clock negativo proveniente da questa uscita interromperà il microcomputer.

«Il flag esterno 1» avrebbe gli ingressi $\overline{\text{CLR-1}}$ e $\overline{\text{PR-1}}$ e le uscite STAT-1 e $\overline{\text{INT-1}}$. Considerazioni analoghe si applicano agli altri flag.

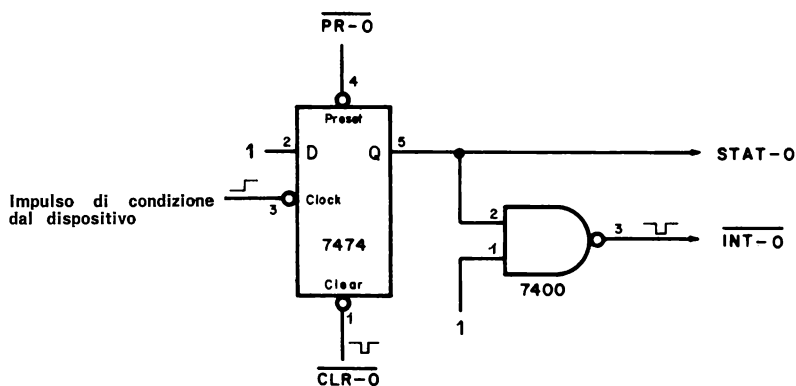



Fig. 8-11. Circuito del flag esterno 0.

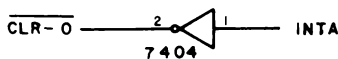



Fig. 8-12. Flag esterni, quale è il flag 0, possono essere associati tramite il segnale INTA.

Come si Azzera un Flag Esterno

Il modo ovvio per azzerare un flag di interrupt è quello di usare il bit di stato INTA, che va a livello logico 1 all'inizio di un ciclo macchina, quando vi è una condizione d'interruzione. Questo stato di livello logico 1 può essere invertito ed applicato all'ingresso $\overline{\text{CLR-0}}$ del flag esterno 0 o di qualunque altro flag, come mostra schematicamente la Fig. 8-12.

L'operazione di azzeramento può talvolta avvenire molto in fretta, in appena 2 μs. Un valore più comune è 7 μs. Non considereremo tutte le sottigliezze inerenti questo tipo di operazione di azzeramento, ma le variabili chiavi sono le istruziooni che vengono usate ed i cicli macchina interni al ciclo istruzioni, nel momento in cui avviene l'interruzione. E' consigliabile di solito fornire una capacità di azzeramento manuale da pannello frontale di uno strumento interfacciato. Un esempio di come fare è dato nella Fig. 8-13. Notate che sia il segnale INTA che il generatore d'impulsi (pulser), possono applicare un impulso di azzeramento al flag. Naturalmente, un gate AND 7408 può essere sostituito dalla combinazione di un gate NAND e di un invertitore 7404.

Un tipo di circuito di azzeramento del flag ugualmente interessato è quello in cui l'impulso di azzeramento è generato da software, cioè tramite un impulso di selezione dispositivo di uscita. E' necessario un impulso di selezione dispositivo negativo, e questo si nota dal simbolo $\overline{\text{DS}}$, dove c'è una sbarretta che indica l'inversione sulle lettere DS. Nella Fig. 8-14 si vedono i circuiti che impiegano un segnale di azzeramento di questo tipo.

Interrupt Differiti e Immediati

I termini interrupt differito e interrupt immediato si possono così definire:

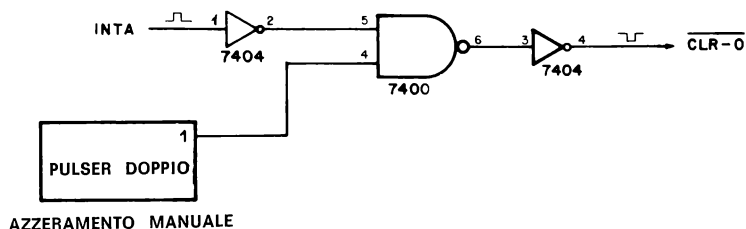



Fig. 8-13. Un circuito che permette sia un ingresso manuale che un ingresso di riconoscimento all'ingresso di azzeramento del flag esterno 0.

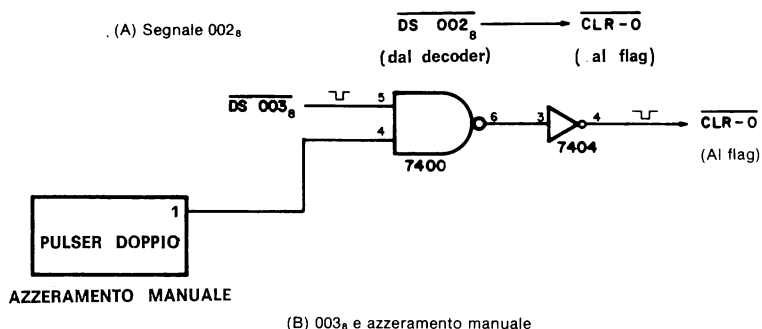



Fig. 8-14. Uso degli impulsi di selezione dispositivo per azzerare il flag esterno 0.

Deffered interrupt (Interrupt differito)	Un interrupt che si verifica in un certo momento, dopo che un flag esterno è stato settato.
Immediate interrupt (Interrupt immediato)	Un interrupt che si verifica non appena un flag esterno viene settato.

Un esempio di interrupt immediato è quando il segnale dell'interruzione viene applicato direttamente all'ingresso di interrupt dell'8080 (pin 14).

Per il flag esterno 0, non appena STAT-0 va a livello logico 1, INT-0 va a livello logico 0 e il microcomputer viene interrotto.

Nella Fig. 8-15 vi mostriamo un semplice circuito d'interrupt differito. La ragione per cui il circuito è «differito», è che potete controllare quando $\overline{INT}$ va a livello logico 0 tramite il segnale di gating DS 002_8.

Naturalmente, questo segnale è generato dal software del microcomputer, quindi può passare un bel po' di tempo tra il momento in cui Q va a livello logico 1, e quando l'uscita del gate NAND

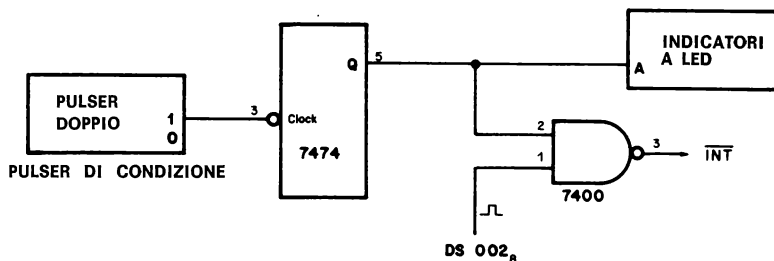



Fig. 8-15. Un circuito di interrupt differito. $\overline{INT}$ non passa a livello logico 0 fino a che Q non è a livello logico 1 e non viene applicato un impulso di selezione dispositivo al gate NAND 7400.

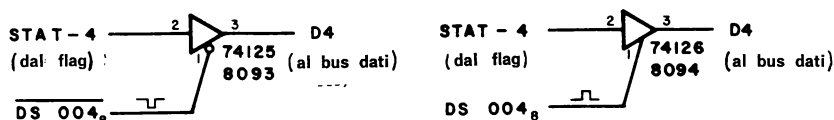



Fig. 8-16. L'uso dei buffer three-state per inserire il bit di stato nell'8080.

7400 va a livello logico 0. Lo vedrete in un altro esempio di questo capitolo. Questo circuito è un esempio di come il software può sostituire l'hardware. Con l'impulso di selezione dispositivo DS 002_8, potete controllare il ritardo di tempo. Questo circuito è particolarmente utile nei sistemi di interruzione a interrogazione ciclica; l'impulso di selezione dispositivi funge da segnale d'interrogazione ciclica.

Uscita del Flag Esterno

Con l'uscita STAT-0 del flag esterno, non si possono mettere in atto troppi espedienti come per il flag esterno 0. Di solito, si inserisce questo bit nel microcomputer. Quindi, per l'uscita STAT-4 del flag esterno 4, potete usare uno qualunque dei due circuiti three-state mostrati nella Fig. 8-16. Questi circuiti impiegano buffer three-state quali il 74125 o il 74126. Vi ricorderete che abbiamo parlato di questi buffer nel Capitolo 7 del Bugbook II. Un impulso di selezione dispositivo, generato da un'istruzione IN, viene usato per inserire i dati nell'accumulatore. Nella Fig. 8-17 vengono fornite le configurazioni dei pin dei due chip.

Se volete inserire le uscite STAT di molti flag esterni, gli autori vi raccomandano di usare un buffer-latch three-state 8212. (Fig. 8-18). Un circuito di questo tipo è molto utile nell'interrupt a interrogazione ciclica. Una volta inseriti gli stati dei livelli logici dei flag esterni nel microcomputer, potete eseguire delle manipolazioni logiche e delle rotazioni per determinare quali flag sono a

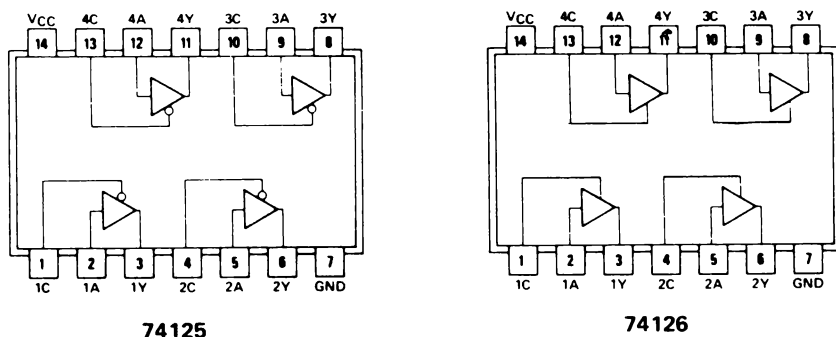
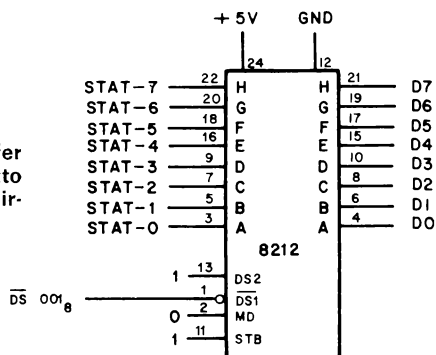



Fig. 8-17. Configurazione dei pin dei buffer three-state 74125 e 74126. Per il chip 74125 un livello logico 0 abilita il buffer; nel 74126 è richiesto un segnale di abilitazione di livello logico 1.

Fig. 8-18. Un chip 8212 usato come buffer d'ingresso per le uscite di stato di otto diversi flag esterni. Questo tipo di circuito si chiama registro di stato.

livello logico 1, quali a livello logico 0, quali hanno cambiato da livello logico 0 a 1, e quali da 1 a 0. Potete usare una qualunque di queste condizioni o cambiamenti di stato di livello logico per richiamare le subroutine che servono i dispositivi legati ai flag esterni.

Infine, ecco come si può elencare la sequenza di eventi che hanno luogo in congiunzione con un interrupt:

- L'uscita del flag esterno, STAT, va a livello logico 1.
- Immediatamente (in caso di interrupt immediato) o dopo un certo ritardo di tempo (in caso di interrupt differito), viene applicato un livello logico 0 al terminale di interrupt, $\overline{INT}$.
- Viene generato un impulso dal microcomputer per riconoscere l'interruzione ($\overline{INTA}$). Questo impulso si può usare per forzare un'istruzione ad un solo byte nel microprocessore e, se lo si desidera, anche per azzerare il flag.
- Il flag esterno viene azzerato non appena riceve l'impulso di azzeramento. STAT ritorna a livello logico 0 e $\overline{INT}$ a livello logico 1.

L'istruzione vettore di ripristino, RST, n, viene applicata al bus di dati solo durante il tempo di riconoscimento dell'interruzione (INTA). *A differenza di tutti gli altri byte d'ingresso, il byte dell'istruzione di ripristino a 8 bit va direttamente al registro istruzioni interno al microprocessore 8080, dove viene eseguito.* Parleremo delle istruzioni di ripristino in un esempio successivo.

Interrupt ad una Sola Linea (Interrupt a Interruzione Ciclica)

Se avete solo un singolo terminale di interrupt e non siete in grado di usare la possibilità di interrupt vettorizzato del microcomputer 8080, potete cablare il circuito mostrato nella Fig. 8-19. Potete interrogare ciclicamente i quattro flag esterni con l'aiuto di quattro diversi impulsi di selezione dispositivo, su ognuno dei quali viene effettuato un gate con un flag esterno, nel modo che mostriamo nella Fig. 8-20.

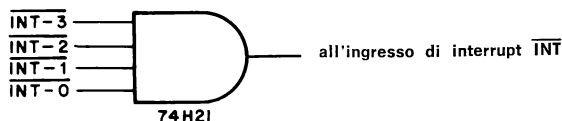



Fig. 8-19. Se uno qualunque degli ingressi di interrupt, INT-n, del gate AND 72H21 passa a livello logico 0, dal gate viene messo in uscita un segnale di interrupt.

MASCHERAMENTO DI INTERRUPT

Nell'operazione di interrupt a interrogazione ciclica, di solito leggete i livelli logici dei flag esterni con l'aiuto di un registro di stato (Fig. 8-18), che inserisce i bit nell'accumulatore del microcomputer. Una volta che i bit sono nell'accumulatore, sorgono le domande: Che cosa fare dopo? Ci sono molte possibilità:

- Far ruotare ogni bit, a turno, nel flag di carry ed impiegare un'istruzione di richiamo condizionato che dipende dal fatto che il flag di carry sia a livello logico 0 o 1. In questo modo, interrogate ciclicamente lo stato di ogni flag esterno e siete in grado di richiamare la subroutine richiesta, se un dato flag è settato.
- *Mascherare* ogni parola di lettura a 8 bit con una parola *«maschera»,* che fondamentalmente rende facile determinare se un dato flag esterno è settato o azzerato. Una volta che l'operazione di mascheramento è stata completata, si possono usare diverse operazioni logiche o aritmetiche per determinare in modo immediato se il flag esterno è settato o azzerato. Una volta fatto questo, si può usare un'istruzione di richiamo condizionato. Di solito è un'operazione di AND che esegue il mascheramento.

Il termine *«maschera»* può così definirsi:

Mask (Maschera) — Una tecnica logica in cui determinati bit di una parola vengono cancellati o inibiti.[4]

E' utile dare un esempio di un'*operazione maschera.* Supponiamo che esistano otto diversi flag esterni che sono collegati ai seguenti dispositivi:

Numero dei flag	*Dispositivo*
0	Telescrivente
1	Lettore a nastro
2	Display CRT
3	Minicomputer
4	Registratore di temperatura
5	Voltmetro digitale
6	Indicatore di pressione
7	Non assegnato

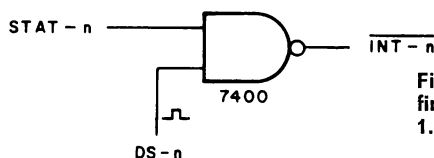



Fig. 8-20. Il segnale $\overline{\text{INT-n}}$ non è generato finché sia STAT-n che DS-n non sono ad 1. Ciò vi permette di effettuare il «polling» dei flag esterni da software.

Ogni dispositivo ha un proprio flag esterno, e tutti e otto i flag sono collegati al registro di stato 8212.

Se vogliamo determinare se il flag del minicomputer è settato o meno, cioè, a livello logico 1, possiamo usare il seguente programma:

Indirizzo di memoria LO	*Istruzione ottale*	*Codice mnemonico*	*Commento*
.	.		.
.	.		.
.	.		.
150	333	IN	Genera un impulso di selezione dispositivo che inserisca i bit dei flag al registro di stato nell'accumulatore
151	001	001	Codice dispositivo per il registro di stato
152	346	ANI	Esegui un'operazione di AND sui contenuti dell'accumulatore con la seguente parola maschera
153	010	010	Parola di maschera che maschera tutti i bit dei flag, eccetto il bit del flag 3
154	302	JNZ	Se i contenuti dell'accumulatore non sono zero, salta alla posizione di memoria data dai due byte d'indirizzo seguenti. Altrimenti, ignora questa istruzione
155	000		Byte d'indirizzo LO
156	002		Byte d'indirizzo HI
.	.		.
.	.		.
.	.		.

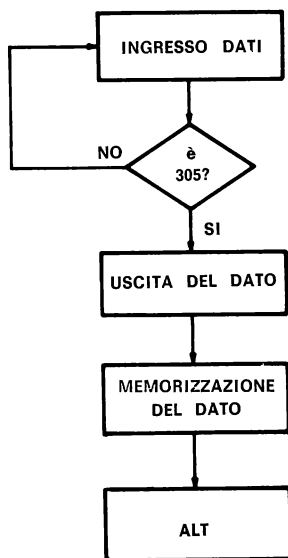
Presumibilmente c'è una routine di servizio alla posizione di memoria H = 002 e L = 000 per trattare la situazione quando il flag esterno per il minicomputer è settato.

INTERFACCIAMENTO CON UNA TASTIERA

In questo paragrafo parleremo di come interfacciare una tastiera ASCII. Alcuni degli schemi mostrati verranno semplificati per chiarezza.

Fate riferimento all'esempio della tastiera del Capitolo 7, quando inserivate un codice di tastiera e lo confrontavate con il codice ASCII per la lettera E. Il diagramma di flusso era come quello mostrato nella Fig. 8-21. Il programma resta in un loop d'ingresso

molto stretto, anche quando non viene premuto nessun tasto e non vi è nessun codice ASCII alla porta d'ingresso 004. Un programma di questo genere è inefficiente per il fatto che spreca il tempo utile del microcomputer, che nel frattempo potrebbe eseguire compiti più significativi.

Fig. 8-21. - Diagramma di flusso che testa un carattere d'ingresso per determinare se è o meno il carattere ASCII E. Quando viene finalmente rivelata una E, il programma la mette in uscita, e poi si arresta.

Come si Testano i Bit dei Flag

La maggior parte delle tastiere ASCII generano un impulso di uscita breve, che si chiama READY o VALID, ogni volta che viene premuto un tasto sulla tastiera. Nel caso di una tastiera, questo impulso ha una durata di 1 μs, un'ampiezza di impulso che è troppo breve per essere inserita direttamente nel microcomputer.

La ragione è che la maggior parte dei loop di timing che testano un solo bit d'ingresso, consumano almeno 12 μs per un microcomputer a 2 MHz. Per un impulso d'ingresso di 1 μs, la stranezza è che per il 92 per cento del tempo, esso non viene rilevato. La soluzione per questo problema consiste nel collegare il segnale di uscita VALID dalla tastiera ad un flag esterno, la cui uscita viene inserita e testata dal microcomputer. Nella Fig. 8-22 vedete un circuito semplificato. Il flip-flop 7474 è il flag, che viene portato ad uno stato logico di uscita 1, dal segnale di 1 μs della tastiera. Il programma testa il livello logico dell'uscita di flag, azzera il flag con impulso di selezione dispositivo $\overline{\text{OUT 065}}$, ed inserisce i dati della tastiera nell'accumulatore. Il programma è il seguente:

Indirizzo di memoria LO	*Istruzione ottale*	*Codice mnemonico*	*Commento*
000	333	IN	Inserisci i dati dei flag dal registro di stato 8212
001	017	017	Codice dispositivo per il registro di stato 8212
002	346	ANI	Maschera i contenuti dell'accumulatore con il byte di dati seguente
003	010	010	Byte di dati di maschera per il bit d'ingresso D3
004	312	JZ	Se il risultato dell'operazione di mascheramento è zero, salta a HI = 000 e LO = 000 e testa nuovamente il flag
005	000		Byte d'indirizzo LO
006	000		Byte d'indirizzo HI
007	323	OUT	Azzera il flag, inviando un impulso di selezione dispositivo all'ingresso CLR
010	065	065	Codice dispositivo per l'ingresso CLR del flag
011	333	IN	Inserisci i dati della tastiera nell'accumulatore
012	005	005	Codice dispositivo per la parte d'ingresso 8212
013			
014			

Il diagramma di flusso di questo programma è fornito nella Fig. 8-23.

Il bit del flag D3 può essere provato in altri modi, come facendolo ruotare nel bit di carry e provando il bit di carry con un'istruzione JC o JNC. Vi sono molti modi diversi di provare i bit di stato o dei flag.

Quando il sistema tastiera/microcomputer viene dapprima attivato, è importante *inizializzare* il sistema azzerando tutti i flag. Tradizionalmente questo è stato fatto usando uno o più interruttori manuali RESET, su cui viene effettuato un gate con gli impulsi di selezione dispositivi come $\overline{\text{OUT 065}}$. Un'alternativa all'uso degli interruttori manuali, è un breve programma di inizializzazione che genera tutti gli impulsi di azzeramento necessari, compreso $\overline{\text{OUT 065}}$.

Interruzioni Vettorizzate

Anche se il programma di prova del flag VALID è più efficiente del programma del Capitolo 7, che inserisce semplicemente i dati della tastiera, la maggior parte dei microcomputer ha cose ben più importanti da fare, che non visualizzare un solo bit. In alcuni casi, il microcomputer esegue un complesso calcolo matematico che richiede moltissimo tempo per l'elaborazione. Il punto è che la maggior parte dei microcomputer dovrebbe essere interfacciata in modo tale da rispondere solo quando viene premuto un tasto sulla tastiera; in tutti gli altri momenti, la tastiera viene ignorata. Un buon dattilografo batterà da cinque a dieci caratteri al secondo,

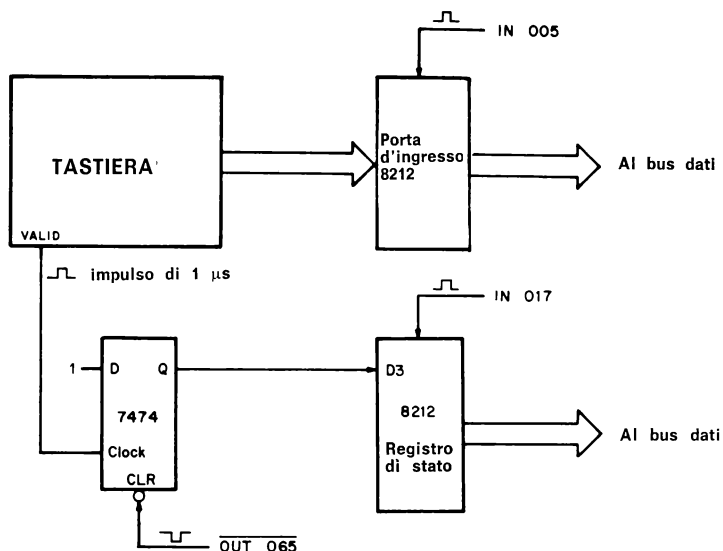



Fig. 8-22. Circuito semplificato che mostra come il flag VALID della tastiera viene testato dal microcomputer.

o da 100 a 200 ms per ogni carattere ASCII. Questa è una velocità molto bassa per i microcomputer standard, e il microcomputer può fare molte altre cose fra una battuta e l'altra. Comunque, una volta che è stato premuto un tasto, sarebbe utile per il microcomputer rispondere immediatamente. Con un microcomputer 8080, la risposta «immediata» può essere effettuata usando *gli interrupt vettorizzati.*

L'interrupt vettorizzato è stato già definito in questo capitolo come un sistema d'interruzione in cui l'interrupt provoca un salto diretto a quella parte del programma che serve l'interrupt stesso. L'insieme di circuiti necessari per l'esempio della nostra tastiera, è mostrato nella Fig. 8-24. Osservate che la porta d'ingresso 8212 resta la stessa, ma che l'impulso VALID è ora inserito nell'ingresso INTERRUPT del nostro microcomputer 8080. In questo caso, l'ingresso di interrupt richiede un impulso di interrupt negativo; se inserite direttamente nel microprocessore 8080, avrete bisogno di un impulso di interrupt positivo, INT.

Una volta che il chip 8080 riceve un impulso di interrupt, e se il flag di interrupt interno al chip è stato già abilitato da un'istruzione di abilitazione all'interrupt, EI, *l'8080 completa l'esecuzione della istruzione, $\overline{INTA}$, che si usa per effettuare un gate su di una istruzione di ripristino ad un solo byte, RST n, direttamente nel registro istruzioni interno al chip 8080.* Questo è l'unico momento in cui potete, usando un buffer three-state esterno, entrare direttamente nel *registro istruzioni,* invece che nell'accumulatore o in

Fig. 8-23. Flow-chart del programma che testa il flag VALID. Quando VALID = 1, un singolo carattere è accettato dalla tastiera.

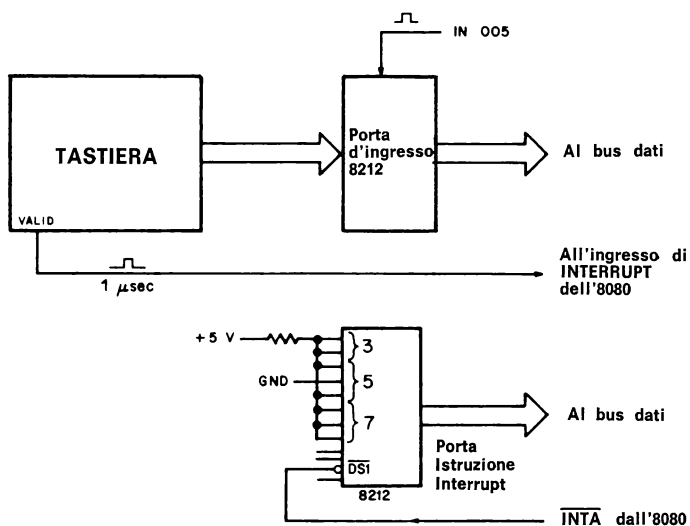



Fig. 8-24. Interrupt vettorizzato semplificato, per la tastiera a caratteri ASCII. Se vi sono altri dispositivi interrompenti nel sistema, bisogna usare un gate NAND prima dell'ingresso INTERRUPT sul chip 8080.

un altro registro universale. In effetti, portate un byte di istruzione nel registro istruzioni. L'hardware usato per fare questa operazione è mostrato nella Fig. 8-24. Esso consiste di un driver a 8 bit gated, che viene chiamato interrupt instruction port. Nella Fig. 8-24 l'istruzione forzata è un'istruzione RTS 5, o **357**, che fa sì che il microcomputer vara immediatamente alla subroutine posizionata a HI = 000 e LO = 050.

Prendiamo in esame il software necessario per un sistema di interrupt vettorizzato. Dato che RST n è un'istruzione di richiamo di subroutine, dovete sistemare uno stack pointer nella memoria di lettura/scrittura. Scrivete poi un'istruzione di abilitazione all'interrupt EI, per permettere al microprocessore 8080 di essere interrotto da un segnale esterno applicato al pin d'ingresso INT. Quindi, saltate ad un'area di memoria, che chiameremo MAIN TASK. Questo è il programma principale, più probabilmente posizionato nella ROM o nella EPROM, che verrà periodicamente interrotto. MAIN TASK può essere posizionato in qualunque punto della memoria, ma gli autori raccomandano di posizionarlo lontano dall'area di memoria della routine di servizio interruzione, che inizia a HI = 000 e LO = 000 e continua circa fino a HI = 000 e LO = 077. Notate, comunque, che la posizione finale della subroutine di interrupt, a LO = 070, può sistemare una subroutine di qualunque lunghezza.

Possiamo riassumere le spiegazioni precedenti mostrando il programma che abbiamo fin qui sviluppato. Quindi:

Indirizzo di memoria LO	*Istruzione ottale*	*Codice mnemonico*	*Commento*
000	061	LXI SP	Posiziona lo stack nella memoria di lettura/scrittura inserendo i due byte d'indirizzo seguenti nello stack pointer interno al chip 8080
001	300	300	Byte d'indirizzo LO
002	003	003	Byte d'indirizzo HI
003	373	EI	Applica il flag d'interrupt interno al chip 8080
004	303	JMP	Salta al MAIN TASK
005	•		Byte d'indirizzo LO di MAIN TASK
006	•		Byte d'indirizzo HI di MAIN TASK
.			
.			
050	333	IN	Inserisci i dati della tastiera nell'accumulatore
051	005	005	Codice dispositivo per la porta d'ingresso 8212
. . . .	Questa sezione di memoria contiene altro software, associato con la routine di servizio della tastiera. E' molto probabile che vi sia un'istruzione di salto verso qualche indirizzo di memoria lontano dall'area della subroutine di servizio interruzione		
057	311	RET	Rientra dalla subroutine; (ultima istruzione nella routine di servizio della tastiera)

Dovrebbe essere chiaro che un'interruzione dalla tastiera costringerebbe il microcomputer a forzare un'istruzione RST 5 nel registro istruzioni. Nell'esecuzione, questo byte istruzioni richiamerà la routine di servizio della tastiera che inizia a HI = 000 e LO = 050. Questa subroutine finisce con una istruzione RET, che permette al microcomputer di ritornare al MAIN TASK, che potrebbe essere sia un semplice loop di controllo che un complicato programma matematico. La routine di servizio della tastiera sarà di solito breve e non consumerà molto tempo.

Il programma precedente funzionerà, ma se tenterete di eseguirlo, incontrerete un certo numero di difficoltà operative. Prima, non sarete in grado di eseguire la routine di servizio della tastiera più di una volta. Perché? Avete sbagliato non riabilitando il flag di interrupt interno al chip 8080. Ricordate la regola seguente:

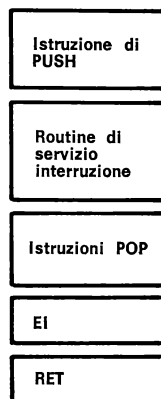
> *Durante un ciclo macchina d'interruzione, il flag di interrupt interno al chip 8080 viene prima disabilitato, poi viene generato un segnale di riconoscimento dell'interruzione,* $\overline{INTA}$*, per permettere ad un'istruzione RST n di essere forzata nel registro istruzioni.*

Il punto chiave è che il flag di interrupt viene disabilitato per evitare ulteriori interruzioni mentre il microprocessore sta servendo l'interruzione in corso. *Se volete riabilitare il flag di interrupt, dovete farlo fornendo un'istruzione EI nella routine di servizio interruzione.* Il flag di interrupt non viene abilitato automaticamente.

E' prassi comune fornire un'istruzione di abilitazione all'interruzione, EI, immediatamente prima dell'istruzioni RET della subroutine di servizio interruzione. Dato che il *flag di interrupt non diventa attivo finché l'istruzione seguente non è stata eseguita,* potete ritornare al MAIN TASK prima che un'altra interruzione venga accettata dal chip 8080.

Se non venisse fornita questa capacità, ci sarebbe il pericolo di riempire troppo la memoria di lettura/scrittura con indirizzi di rientro che aspettano di essere usati, perché le routine di servizio interruzione sono state nuovamente interrotte, prima di avere modo di rientrare.

Sotto tutti gli altri aspetti, trattate le vostre subroutine di vettore come subroutine normali. Se i contenuti dei registri sono importanti, usate le istruzioni PUSH e POP per conservare e rimemorizzare i registri. La Fig. 8-25 mostra come apparirebbe una tipica subroutine d'interruzione. Dato che esistono solo sette posizioni di memoria fra il nostro indirizzo vettore a LO = 050 e l'indirizzo vettore seguente a LO = 060, non sarete in grado di farci stare le vostre quattro istruzioni PUSH, le quattro istruzioni POP, la EI e la RET, e le istruzioni di servizio della tastiera senza incrociare la o le posizioni seguenti della subroutine di vettore. Mettete

Fig. 8-25. Una tipica subroutine d'interruzione. Le prime istruzioni, le istruzioni PUSH, conservano lo stato del microcomputer. Vicino alla fine della subroutine, lo stato del microcomputer viene ripristinato nella locazione originale.

invece un'istruzione di salto a LO = 050, 051 e 052, che trasferisce il controllo ad un'area di memoria dove avreste più spazio per il software. Alla fine della routine di servizio, l'istruzione RET rimanderà ancora il controllo di programma al punto in cui il MAIN TASK era stato interrotto. La relazione fra il MAIN TASK, l'istruzione di salto a HI = 000 e LO = 050 e il software di servizio della tastiera è mostrata nella Fig. 8-26. La creazione di un'interfaccia per la tastiera avrebbe potuto essere fatta in modo più complesso. Per esempio, si sarebbero potuti usare gli interrupt differiti o gli interrupt con priorità. Nell'esempio precedente, c'era poco incentivo a farlo.

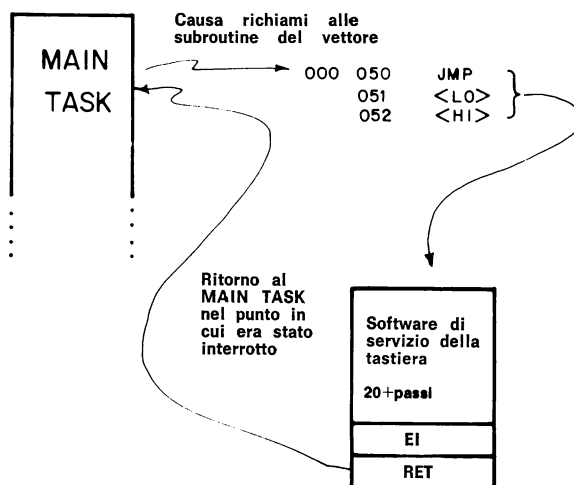



Fig. 8-26. Relazione fra il MAIN TASK, il salto alla subroutine di vettore, ed il software di servizio della tastiera, che è posto in un altro punto della memoria.

INTERRUPT CON PRIORITA'

Gli interrupt con priorità sono posti in ordine di importanza, in modo che alcuni dispositivi interrompenti hanno la precedenza su altri. Vengono usati in qualunque momento si verifichi un certo numero di interrupt nello stesso momento, o in qualunque momento ci sia bisogno di determinare quali dispositivi interrompenti siano più importanti.

Due vari modi di stabilire la priorità, quello più semplice è definire la priorità da software e poi interrogare ciclicamente i dispositivi interrompenti e determinare quali dispositivi dovrebbero essere serviti ed in quale ordine. Per esempio, consideriamo il circuito della Fig. 8-27. Si vedono tre interruzioni, ma in un sistema reale se ne potrebbero includere molte altre. Un'interruzione si verifica in qualunque momento uno dei flip-flop dei flag viene

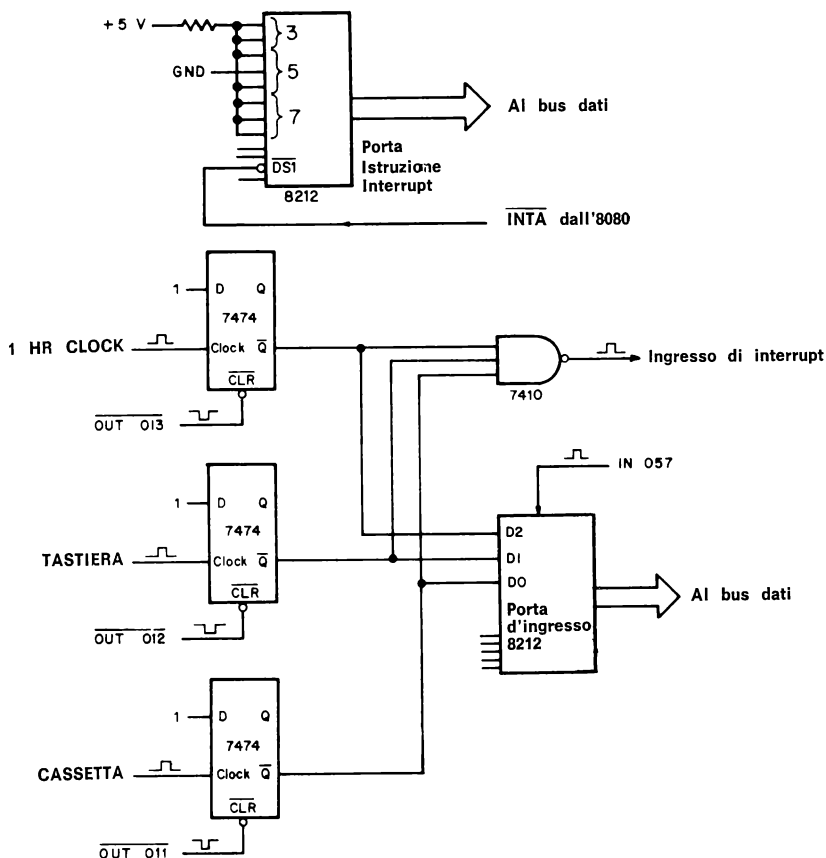



Fig. 8-27. Circuito d'interrupt a interrogazione ciclica, che consiste di tre dispositivi interrompenti. L'impulso di interrupt positivo è collegato direttamente al pin d'ingresso INT sul microprocessore 8080.

settato da un impulso, al suo ingresso di clock. Noi lo mostriamo come un impulso di interrupt positivo, che è quanto richiesto dalla maggior parte dei microcomputer 8080. Se l'impulso d'interrupt viene applicato direttamente al chip 8080, si dovrebbe usare un chip 7410, come mostra la Fig. 8-27. Quando l'8080 accetta l'interruzione, genera un impulso $\overline{\text{INTA}}$ che attiva un gate sul codice istruzione RST, 357, che viene cablato in precedenza alla porta dell'interruzione, direttamente nel registro istruzioni. Nel corso dell'esecuzione dell'istruzione RST 5, viene richiamata una subroutine a HI = 000 e LO = 050. L'istruzione 5 provoca un vettore, o un salto, alla subroutine indicata, dove ha luogo l'interrogazione ciclica da parte del software, dei dispositivi interrompenti.

Tutti i flag, quello del clock da un'ora, quello della tastiera, e della cassetta, mostrati nella Fig. 8-27, vengono inseriti a livello logico 1 nel gate NAND a tre ingressi 7410, se non è richiesto servizio, e come livello logico 0, se il servizio è richiesto.

Il normale livello logico dell'uscita del gate 7410 è 0; è necessario un impulso di clock positivo a questa uscita per interrompere la maggior parte dei microcomputer 8080. Nel programma che segue, la priorità è stabilita in modo tale che la cassetta ha la priorità più alta (dispositivo ad alta velocità), la tastiera viene dopo in priorità (dispositivo a bassa velocità) e il clock da un'ora è l'ultimo (dispositivo incredibilmente lento). Ecco il software:

Indirizzo di memoria LO	*Istruzione ottale*	*Codice mnemonico*	*Commento*
050	333	IN	Inserisci i bit di stato dei tre flag
051	057	057	Codice dispositivo per la porta d'ingresso 8212
052	057	CMA	Completa i bit di stato (1 → 0 e 0 → 1)
053	346	ANI	Maschera tutti i bit eccetto i bit D0, D1 e D2
054	007	007	Byte di maschera
055	037	RAR	Fai ruotare i contenuti dell'accumulatore a destra attraverso il carry (il bit D0 viene fatto ruotare nel bit di carry)
056	332	JC	Se il bit di carry è a livello logico 1, salta alla routine di servizio cassetta posizionata a HI = 003 e LO = 100
057	100		Byte d'indirizzo LO della routine di servizio cassetta
060	003		Byte d'indirizzo HI della routine di servizio cassetta
061	037	RAR	Fai ruotare i contenuti dell'accumulatore ancora una volta verso destra (il bit D1 viene fatto ruotare ora nel bit di carry)
062	332	JC	Se il bit di carry è a livello logico 1, salta alla routine di servizio tastiera posizionata a HI = 003 e LO = 200
063	200		Byte d'indirizzo LO della routine di servizio tastiera

064	003		Byte d'indirizzo HI della routine di servizio tastiera
065	037	RAR	Fai ruotare i contenuti dell'accumulatore verso destra (il bit D2 viene ora fatto ruotare nel bit di carry)
066	332	JC	Se il bit di carry è a livello logico 1, salta alla routine di servizio clock da un'ora, posizionata a HI = 003 e LO = 300
067	300		Byte d'indirizzo LO della routine di servizio clock da un'ora
070	003		Byte d'indirizzo HI della routine di servizio clock da un'ora
071	166	HLT	Se siete arrivati a questo punto, c'è qualcosa che non va. Il microcomputer si è fermato; scoprite qual'è il problema

Notate l'uso dell'istruzione CMA a LO = 052. Normalmente, il flag sarebbe a livello logico 1, se il servizio è richiesto, e a livello logico 0, se il servizio non è necessario. Nell'esempio della Fig. 8-26 la situazione è esattamente opposta. L'uso dell'istruzione CMA mostra come sia facile invertire otto bit di dati dell'accumulatore.

L'istruzione ANI a LO = 053 maschera i bit da D3 a D7, illustrando così come si può usare un'operazione di mascheramento. Comunque, dato che vengono usate le istruzioni RAR, e dato che teniamo conto di quante volte abbiamo fatto ruotare i contenuti dell'accumulatore, l'istruzione ANI qui non è proprio necessaria. Si potrebbero implementare nel software altri metodi di test dei bit. Queste routine di servizio sono molto simili a quelle usate con le normali routine di servizio, come mostra la Fig. 8-25. Queste routine terminano generalmente con un'istruzione di abilitazione all'interruzione, EI, e con un'istruzione di rientro, RET. Anche quando si usa l'interrogazione ciclica, dobbiamo tornare ancora al programma principale che era stato interrotto.

La suddetta routine di interrogazione ciclica passa attraverso gli indirizzi di vettore LO = 060 e LO = 070. Questo non è un errore; non esistono altri interrupt che usano questi indirizzi vettori.

Sono possibili altre variazioni nel programma di interrogazione ciclica. Dato che i flag di interrupt vengono inseriti nell'accumulatore, potremmo voler conservare i contenuti dell'accumulatore e dei flag al momento in cui il MAIN TASK è stato interrotto. Per farlo, forniremo un'istruzione PUSH PSW a LO = 050. Ogni subroutine di servizio richiederebbe un'istruzione POP PSW immediatamente prima dell'istruzione EI. Sarebbero necessarie altre istruzioni PUSH e POP per conservare i contenuti dei registri. Infine, è molto probabile che la prima istruzione di ogni routine di servizio azzererebbe il flag associato con la subroutine. Quindi, come mostra la Fig. 8-27, un impulso $\overline{\text{OUT 011}}$ azzererebbe il flag della cassetta, un $\overline{\text{OUT 012}}$ azzererebbe il flag della tastiera, e un $\overline{\text{OUT 013}}$ azze-

rerebbe il flag del clock di un'ora.

Gli autori trovano che il programma suddetto di interrogazione ciclica sia molto utile nel loro lavoro. Andranno ugualmente bene altri schemi software di interrogazione ciclica, ma quello dato nell'esempio è semplice ed efficace.

Il clock di un'ora solleva una domanda importante: perché costruire un clock hardware esterno di un'ora quando avete già del software per fare la stessa cosa all'interno, con meno di 50 byte di istruzione? La risposta a questa domanda dipende da come usate il vostro computer. Se il vostro computer può soffermarsi sul software del clock da un'ora e non fare niente altro, o se state usando le interruzioni e potete tollerare una piccola quantità di errori nella subroutine a ritardo di tempo di un'ora (dovuto al tempo che il microcomputer perde per le interruzioni) allora usate il software.

Se avete bisogno di sapere il tempo esatto, usate l'hardware. Ricordate, quando usate le interruzioni e le routine di servizio interruzione, state essenzialmente aggiungendo dell'altro software al flusso di programma del MAIN TASK. Questo software addizionale occupa del tempo. Dovete non solo controllare il dispositivo interrompente, ma anche servirlo. Se interrompete una routine a ritardo di tempo software di un'ora per quattro volte con una routine di servizio dispositivi di due minuti, l'esecuzione della routine a ritardo di tempo occuperà un'ora e otto minuti invece di un'ora. Le operazioni della routine software di un'ora vengono sospese quando c'è un'interruzione che richiede che vengano eseguiti altri compiti. Il clock esterno di un'ora si può chiamare clock a tempo reale, dato che esso occupa del tempo reale, in opposizione al tempo software del microcomputer.

INTERRUPT CON PRIORITA' DA HARDWARE

Gli interrupt con priorità si possono generare anche usando l'hardware. Gli interrupt con priorità da hardware sono molto importanti quando un certo numero di dispositivi interrompenti sono collegati ad un microcomputer e richiedono tutti un servizio relativamente veloce. «L'espediente» usato è semplice: *tutti i dispositivi interrompenti generano la loro istruzione di ripristino, RST n, che, quando è inserita nel microcomputer 8080, provoca un vettore immediato alla posizione HI = 000 e LO = 0N0, dove N può essere 0, 1, 2, 3, 4, 5, 6 o 7.* Inoltre, la priorità viene assegnata automaticamente dall'hardware, in modo che il dispositivo 7 ha priorità maggiore del dispositivo 6, che ha priorità maggiore del dispositivo 5, ecc. In altre parole, se $>$ rappresenta la priorità, allora:

$$7 > 6 > 5 > 4 > 3 > 2 > 1 > 0$$

Il circuito che dovreste usare è mostrato nella Fig. 8-28.

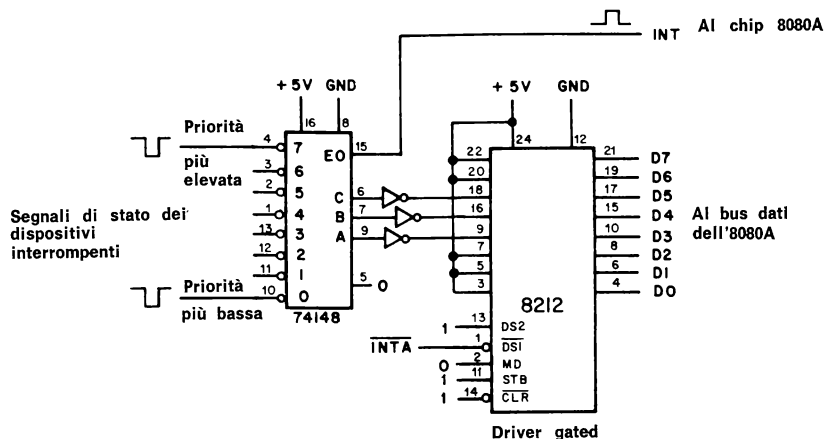



Fig. 8-28. Circuito d'interrupt con priorità hardware che genera otto diverse istruzioni di ripristino, RST n, che hanno priorità 7 > 6 > 5 > 4 > 3 > 2 > 1 > 0.

Il circuito integrato codificatore di priorità, da otto a tre linee, 74148, è un chip a 16 pin la cui configurazione dei pin e tabella della verità è mostrata nella Fig. 8-29. Notate che gli ingressi e le uscite dei dati sono attivi al livello logico basso. Il chip 74148 accetterà fino a otto ingressi di livello logico 0 dai flag, come quelli mostrati nella Fig. 8-28 e metterà in uscita il *codice binario per l'ingresso numericamente più alto tra quelli a livello logico 0.*

Per esempio, se avete richieste simultanee di interruzione da entrambi i dispositivi 5 e 7, il dispositivo 7 ha il massimo della priorità e il chip 74148 e gli invertitori, forniranno un codice ottale 7 per la cifra ottale N dell'istruzione di ripristino, 3N7. Questo porta il programma alla posizione HI = 000 e LO = 070. L'istruzione RST 0 non viene usata spesso dato che il suo solo effetto è quello di resettare il microcomputer e ridare il via al programma MAIN TASK.

La Fig. 8-28 è stata semplificata per chiarezza. Non si vedono i flag necessari e le linee di azzeramento dei flag. Si potrebbero aggiungere al circuito degli elementi hardware addizionali per renderlo più efficiente. Essi sarebbero un decodificatore addizionale 7442, per generare gli impulsi di azzeramento dei flag senza usare le istruzioni OUT, ed un registro maschera in modo da mascherare o smascherare vari dispositivi, tramite l'hardware esterno. La Fig. 8-30 mostra il circuito modificato. E' uno schema molto sofisticato d'interrupt con priorità, che fornisce grande flessibilità nell'uso delle interruzioni vettorizzate insieme al microcomputer 8080.

Dovete prima decidere a quali dispositivi sarà permesso di interrompere il microcomputer e a quali no. Sviluppate una configura-

SN54148, SN74148
FUNCTION TABLE

INPUTS									OUTPUTS				
EI	0	1	2	3	4	5	6	7	A2	A1	A0	GS	EO
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	X	L	H	H	L	H	L	L	H
L	X	X	X	X	L	H	H	H	L	H	H	L	H
L	X	X	X	L	H	H	H	H	H	L	L	L	H
L	X	X	L	H	H	H	H	H	H	L	H	L	H
L	X	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H

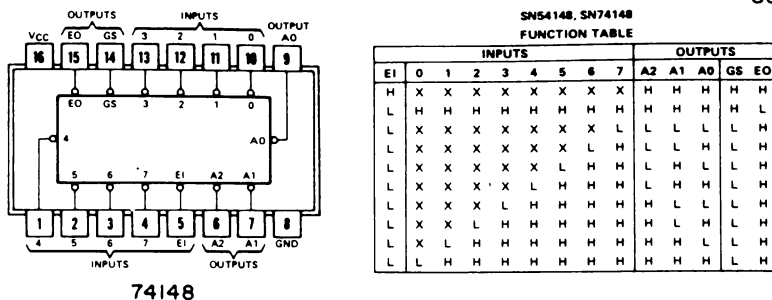



Fig. 8-29. Configurazione dei pin e tabella della verità del codificatore di priorità da otto a una linea 74148.

zione maschera a 8 bit, in cui ai dispositivi d'interruzione è assegnato un livello logico 1 e a quelli che non interrompono un livello logico 0. Questi otto bit vengono messi in uscita dall'accumulatore sui due latch 7475 mostrati a sinistra nella Fig. 8-30. A questo scopo si usa un'istruzione OUT 030. La posizione del bit D7 corrisponde al dispositivo 7, che ha la priorità superiore e può generare un vettore all'indirizzo HI = 000 e LO = 070. Quei dispositivi che vengono mascherati useranno probabilmente un ingresso di registri di stato per richiedere il servizio, come già mostrato in questo capitolo.

Le richieste d'interruzione provenienti dai flag vengono collegate ai gate OR (uno lo vedete a sinistra nella Fig. 8-30) collegati al registro maschera 7475 e le richieste d'interruzione non mascherate vengono passate al latch a 8 bit 74100. In qualunque momento venga abilitato il flag di interrupt interno al chip 8080, INTE è alto ed abilita il chip 74100. Le azioni del decodificatore di priorità 74148 e della porta della istruzione d'interruzione, sono state già descritte. Quando l'interrupt viene ricevuto dal chip 8080, esso disabilita il suo flip-flop interno di abilitazione all'interrupt e l'uscita INTE ritorna a livello logico 0, effettuando un latch su qualunque interruzione presente al chip 74100. Il segnale INTA non solo inserisce il byte istruzioni RST n; genera anche degli impulsi nel decodificatore 7442, per produrre un impulso di azzeramento che azzera il flip-flop associato con l'interrupt che viene servito in quel momento.

Si possono usare molti altri schemi d'interruzione, compresi quelli basati sull'8214 priority interrupt control unit. Il chip programmable interrupt controller 8259 della Intel, può generare otto interrupt con priorità vettorizzate per un microcomputer 8080 e può venir messo in cascata per 64 interruzioni di priorità vettorizzate senza circuiti addizionali. Sebbene gli interrupt permettono una risposta veloce agli eventi esterni o alle richieste di servizio, il loro uso richiede un certo grado di sofisticazione sia nell'hardware che nel software.

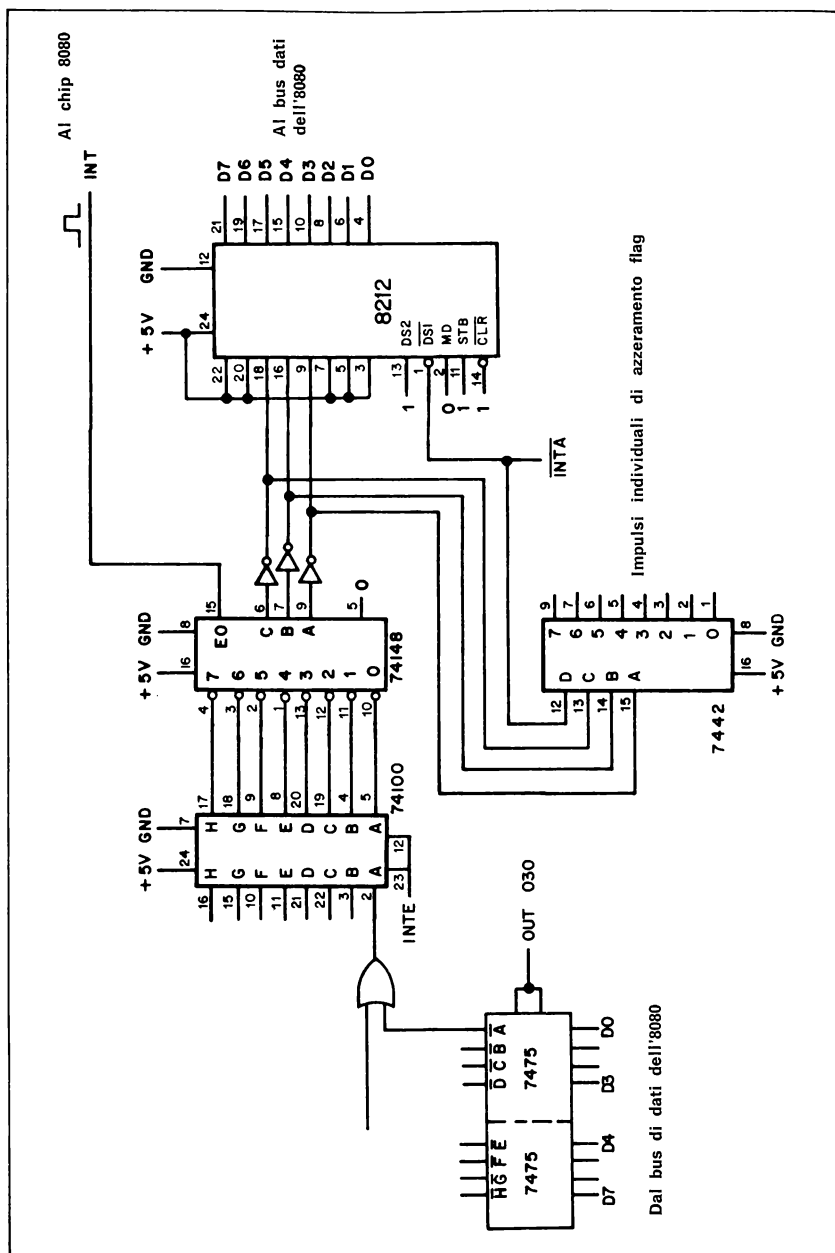



Fig. 8-30. Esempio di un sofisticato circuito di interrupt con priorità che genera impulsi di azzeramento dei flag individuali e che ha un registro maschera, in modo che i dispositivi possono essere mascherati o smascherati tramite l'hardware esterno.

INTERRUPT CON PRIORITA' DA SOFTWARE

Come ultimo argomento di questo capitolo, prendiamo in considerazione il software necessario per un sistema di interrupt con priorità. Supponiamo di avere solo due dispositivi interrompenti: il dispositivo 7 a priorità superiore e il dispositivo 2 a priorità inferiore. Ogni dispositivo genera il suo byte istruzoni di ripristino, che provoca un vettore nella posizione LO = 070 o LO = 020, rispettivamente. Supponiamo anche che il dispositivo a priorità superiore interrompa il programma principale, chiamato MAIN TASK, su una base regolare, e venga servito velocemente tramite il software. Il dispositivo 2 a priorità inferiore si suppone interrompa in maniera irregolare. Per esempio, il dispositivo 2 potrebbe essere un altro microcomputer che spinge blocchi di dati nel nostro microcomputer. Il software più importante è il software del MAIN TASK che viene eseguito in qualunque momento non vengano serviti i dispositivi esterni. Se il software non fosse importante, non sarebbe il «main task» eseguito dal microcomputer. Nel MAIN TASK definiamo lo stack pointer con un'istruzione LXI SP ed abilitiamo anche il flip-flop di interrupt usando un'istruzione EI.

Dato che gli interrupt possono avvenire in qualunque momento, sono necessarie sia le istruzioni PUSH che le istruzioni POP nelle routine di servizio interruzione. Tali istruzioni conserveranno e rimemorizzeranno i registri che vengono alterati nelle routine di servizio. L'esecuzione del software può essere rappresentata graficamente da una linea di tempo, come mostra la Fig. 8-31. Notate che il dispositivo a priorità superiore ha interrotto il MAIN TASK quattro volte, mentre quello a priorità inferiore lo ha interrotto una volta. Il dispositivo a priorità superiore interrompe in modo regolare, come potete vedere dai suoi spazi sulla linea di tempo del MAIN TASK. La linea in neretto indica quando l'interruzione è abilitata.

La Fig. 8-32 mostra una linea di tempo più realistica. La linea di tempo della Fig. 8-31 è alquanto ingannevole dato che fa vedere solo il tempo trascorso nel MAIN TASK.

E' più corretto far notare il tempo reale trascorso sia nel MAIN TASK che nelle subroutine. Nella Fig. 8-32 il MAIN TASK inizia ad operare e viene interrotto dal dispositivo a priorità superiore. Dopo aver eseguito la subroutine di servizio dispositivi a priorità superiore, il controllo è rimandato al MAIN TASK, che viene interrotto più tardi sulla linea di tempo, dal dispositivo a priorità inferiore. Il controllo ritorna finalmente al MAIN TASK che viene quindi interrotto a intervalli ripetuti dal dispositivo a priorità superiore. Chiaramente, ci vuole molto più tempo a raggiungere la fine del MAIN TASK quando viene ripetutamente interrotto da altri dispositivi, ognuno dei quali richiede di essere servito. Durante un periodo di timing, queste interruzioni sarebbero disastrose se si fosse impegnati in loop a ritardo di tempo programmati per generare un opportuno delay.

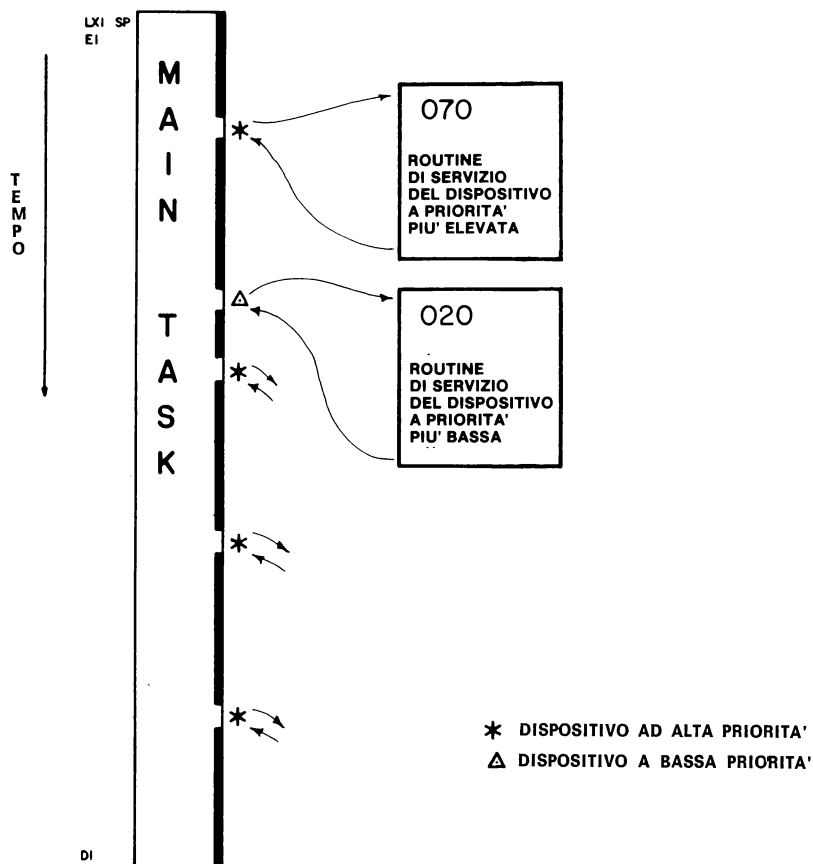



Fig. 8-31. Linea del tempo di esecuzione del programma MAIN TASK. Le interruzioni da parte dei dispositivi a priorità inferiore e superiore sono indicate dai simboli * e △, rispettivamente.

Abbiamo supposto che il dispositivo a priorità superiore interrompa su di una base regolare. Esso ha cercato probabilmente di interrompere l'esecuzione della subroutine del dispositivo a priorità inferiore mostrata nella Fig. 8-32. Se quello alto ha una priorità superio-
Il controllo ritorna finalmente al MAIN TASK che viene quindi interrotto a intervalli ripetuti dal dispositivo a priorità superiore.
re a quello basso, perché non è avvenuta un'interruzione? La risposta è che il flag di interrupt interno al chip 8080 *non era abilitato durante l'esecuzione della subroutine di servizio dispositivi a priorità inferiore.* Nel nostro primo tentativo di scrivere il software di servizio interruzione, abbiamo dimenticato di prevedere in essa questa possibilità. Come risultato, i dati o i segnali provenienti dal dispositivo a priorità superiore andavano persi durante il softwa-

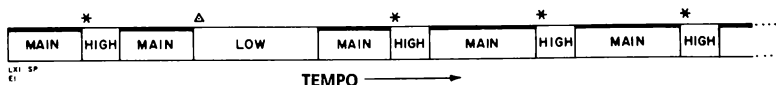



Fig. 8-32. Linea del tempo di esecuzione del programma MAIN TASK e delle subroutine di servizio dispositivi a priorità inferiore e superiore.

re di servizio del dispositivo a priorità inferiore. Possiamo facilmente correggere il nostro software mettendo l'istruzione di abilitazione all'interruzione, EI, all'inizio della subroutine di servizio del dispositivo a priorità inferiore. Possiamo anche progettare dell'hardware per memorizzare i dati o i segnali che hanno luogo durante un'interruzione perduta.

Ponendo l'istruzione di abilitazione all'interruzione, EI, all'inizio della subroutine di servizio dispositivi a priorità inferore, possiamo incontrare un nuovo problema: un flusso software del dispositivo a priorità inferiore «chopped», come mostra la Fig. 8-33. Per metterlo maggiormente in rilievo, abbiamo supposto che il dispositivo a priorità superiore interrompa il dispositivo a priorità inferiore due volte, spezzando il software a priorità inferiore in tre pezzi col dispositivo software a priorità inferiore così suddiviso, dobbiamo chiederci se siamo in grado di completare il software a priorità inferiore *prima che il dispositivo a priorità inferiore generi una nuova interruzione.* E' assolutamente possibile per il dispositivo a priorità inferiore interrompere il microcomputer mentre sta ancora tentando di servire l'ultima richiesta d'interruzione del dispositivo a priorità inferiore. Mentre la risposta all'interruzione è veloce, il tempo di esecuzione reale può essere molto più lento del tempo richiesto per un solo passaggio attraverso il software di servizio interruzione. Questa è una conseguenza del fatto che possiamo interrompere le nostre interruzioni. Tali considerazioni dovrebbero darvi un'idea dell'attenzione necessaria nell'uso degli interrupt con priorità. E' molto facile per un microcomputer diventare *«legato alle interruzioni»,* cioè passare tutto il suo tempo a controllare ed a servire le interruzioni, e non avere più tempo per il software del MAIN TASK.

Nel software del nostro MAIN TASK, potremmo voler evitare che le interruzioni si verifichino durante lo svolgersi di software al tempo o comunque in concomitanza con calcoli complessi dipendenti dal tempo. L'istruzione di disabilitazione all'interruzione, DI, permette al microcomputer di essere immune dalle interruzioni

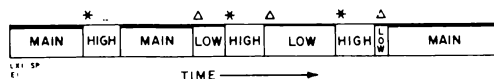



Fig. 8-33. Linea del tempo di esecuzione del programma, che dimostra l'interruzione di una routine di servizio interruzione. Il software d'interruzione «basso» viene interrotto due volte dal dispostivo a priorità superiore.

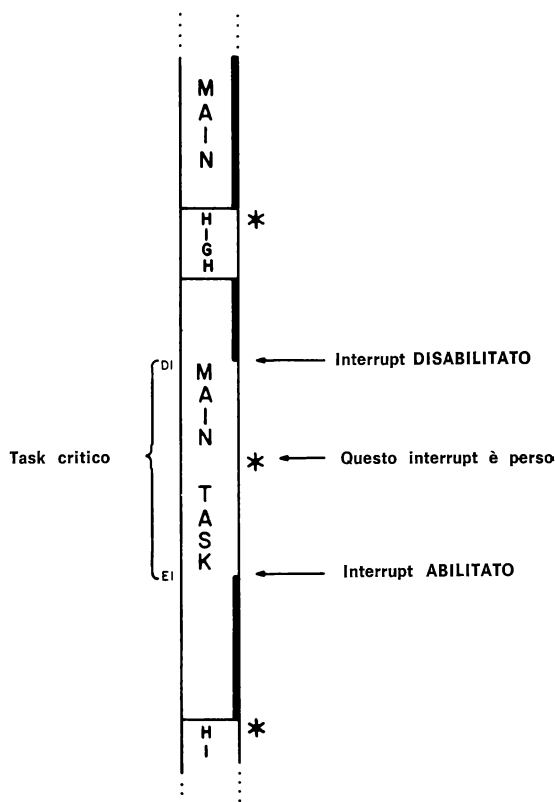



Fig. 8-34. Linea del tempo di esecuzione del programma che dimostra l'uso delle istruzioni DI e EI per permettere che venga eseguita una operazione critica nel MAIN TASK. In questo caso, comunque, viene persa o ritardata un'interruzione mentre l'operazione critica viene eseguita.

esterne. Tale situazione è mostrata nella Fig. 8-34. Il flip-flop di interrupt viene disabilitato per permettere che venga eseguito un compito difficile, e poi ripristinato. Sfortunatamente, la linea di tempo per l'esecuzione del MAIN TASK mostra che è andata perduta un'interruzione da parte del dispositivo a priorità superiore. Senza il supporto di hardware addizionale, di solito complesso, è molto facile perdere segnali o dati dei dispositivi interrompenti, mentre il flag di interrupt è disabilitato. L'importante è *che non sappiamo quando un dispositivo esterno interromperà il MAIN TASK, e non possiamo essere sicuri che ciò non avverrà durante il periodo in cui il flip-flop di interrupt è disabilitato.* Come ovviare a questo problema? Non è facile, e questa è la ragione per cui dobbiamo prestare moltissima attenzione nell'uso delle interruzioni.

Un altro tipo d'interrupt che può interessarci è l'interrupt *a orien-*

tamento di tempo. Si usa solo un'interrupt: il clock interrompente ogni 10 millisecondi, o un altro periodo ragionevole. Quando viene interrotto, il microcomputer usa una tabella di ricerca per determinare quali dispositivi controllare per vedere se richiedono assistenza. Alcuni dispositivi vengono sempre controllati, mentre altri dispositivi più lenti potrebbero essere controllati una volta ogni cento. Questa è una tecnica utile, ma richiede una notevole quantità di software per funzionare bene.

I microprocessori 8080 più recenti permettono alle istruzioni a più byte di essere inserite durante un'interruzione, in modo che potrebbero essere inseriti un richiamo o un salto completo di 3 byte. La possibilità di forzare un'istruzione a tre byte elimina il bisogno di usare le istruzioni di ripristino e le posizioni di vettore associate, e vi fornisce una flessibilità molto maggiore nell'uso dell'hardware e del software. Il controller di interrupt programmabile 8259 della Intel permetterà di eseguire richiami diretti alle subroutine di servizio di interruzione; ma si tratta di un dispositivo complesso, non per principianti.

Concluderemo questo capitolo con alcune raccomandazioni. Le interruzioni sono difficili da mettere a punto. Alcune interruzioni possono aver luogo quasi in qualunque momento, i tipici programmi di messa a punto del software sono difficili da applicare; la maggior parte non sono efficaci. E' necessario uno speciale software diagnostico per provocare le interruzioni nelle applicazioni specifiche. *Se potete evitare di usare le interruzioni, fatelo.* Occupate il vostro tempo utile con altri tentativi di non interruzione, se possibile. I vostri sforzi saranno ricompensati.

TEST

Questo test verifica quanto avete capito delle tecniche di interruzione discusse in questo capitolo. Per favore scrivete le risposte su di un foglio a parte.

8.1 Descrivete in che modo viene caricata la seguente sequenza di istruzioni relative allo stack; PUSH D, PUSH B, PUSH PSW, PUSH H.

8.2 Disegnate un semplice circuito per un flag esterno e spiegate tutti gli ingressi e tutte le uscite.

8.3 Una piccola routine richiede quattro byte di istruzione e viene usata dieci volte in un microcomputer 8080. Spiegate come, e se, decidereste di trasformare la routine in subroutine, completa di istruzione di rientro.

8.4 Descrivete i diversi tipi di istruzioni di subroutine nel set di istruzioni dell'8080.

8.5 Descrivete i diversi tipi di istruzioni di stack nel set di istruzioni dell'8080

8.6 Spiegate la differenza fra i seguenti tipi di interrupt:
Differito
Vettorizzato
Su di una sola lînea
Immediato
A più livelli
A interrogazione ciclica.

La vostra risposta sarà accettabile, se sarete stati in grado di rispondere a tutte le domande suddette correttamente a libro chiuso in 90 minuti.

CHE COSA AVETE REALIZZATO IN QUESTO CAPITOLO?

All'inizio di questo capitolo si era stabilito che, alla fine, avreste dovuto essere in grado di:

- Dare la definizione dei termini: subroutine, SSI, MSI, LSI, allocare, stack, interruzione, interrogazione ciclica, software driver, interrupt vettorizzato, interrupt disabilitato, flag esterno, interrupt differito e registro di lettura.

  Le definizioni di questi termini sono state fornite nel capitolo in vari punti.

- Spiegare come maschereste una parola a 8 bit per ottenere il livello logico del bit 5.

  Come esempio, abbiamo fornito un semplice programma in cui veniva mascherato il bit 3 di una parola a 8 bit. Dovreste essere in grado di applicare le tecniche analoghe al bit 5.

- Spiegare come vengono caricate le informazioni digitali e come vengono prelevate dallo stack del microcomputer 8080.

  Lo abbiamo fatto con l'aiuto della Fig. 8-3, che è un diagramma molto interesssante di un tipico stack.

- Eseguire un calcolo approssimato che vi dirà quando usare una subroutine.

  Vedere la Fig. 8-6 e quanto detto in proposito.

- Descrivere come interfaccereste una tastiera ASCII.

  Questo argomento viene discusso, nei minimi particolari, verso la fine del capitolo.

APPENDICE 1

Riferimenti

I riferimenti bibliografici indicati nel testo sono i seguenti:

1. Charles L. Garfinkel, della Keithley Instruments, Inc., è il creatore di questa definizione.
2. Donald Eadie, *Introduction to the Basic Computer,* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1973.
3. Texas Instruments, Inc., *Microprocessor Handbook,* Dallas, Texas, 1975.
4. Rudolf F Graf, *Modern Dictionary of Electronics,* Howard W. Sams & Company, Inc., Indianapolis, 1977.
5. Microdata Corp., *Microprogramming Handbook,* Santa Ana, Californa, 1971.
6. Abraham Marcus and John D. Lenk, *Computers for Technicians,* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1973.
7. Intel Corp., *Intel 8080 Assembly Language Programming Manual,* Santa Clara, California, 1974.
8. Intel Corp., *Intel Intellec 8/ Mod. 80 Microcomputer Development System Reference Manual,* Santa Clara, California, 1975.
9. Intel Corp., *Intel 8080 Microcomputer Systems User's Manual,* Santa Clara, California, July, 1975.
10. J. Blukis and M. Baker, *Practical Digital Electronics,* Hewlett-Packard Company, Santa Clara, California, 1974.
11. *An Introduction to Microcomputers,* Adam Osborne and Associates, Inc., Berkeley, California, 1975.
12. The Intel literature è disponibile presso la Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051 [telephone: (408) 246-7051], o da rivenditori, rappresentanti e distributori.
13. Le octal card possono essere ottenute dalla Tychon, Inc., P. O. Box 242, Blacksburg Virginia 24060.
14. Charles J. Sippl and David A. Kidd, *Microcomputer Dictionary and Guide,* Matrix Publishers, Inc., Champaign, Illinois, 1976.

APPENDICE 2

Set di Istruzione dell'8080

Questa appendice fornisce un sommario di tutte le caratteristiche che più importanti di ciascun istruzione costituente il set dell'8080A: il numero dei cicli macchina, il numero degli stati, il tipo di indirizzamento di memoria e i flag che risultano modificati dopo l'esecuzione dell'istruzione stessa. Viene fornita la descrizione di ogni istruzione e, in alcuni casi, si presentano esempi del loro utilizzo.

PROGRAMMAZIONE DEL MICROCOMPUTER

Se non avete uno specifico background sui computer oppure una particolare predisposizione per la programmazione, probabilmente troverete la programmazione in linguaggio macchina ed assembler abbastanza noiosa, oltreché difficile, almeno all'inizio. In effetti non vi sono delle scorciatoie da seguire per imparare a programmare. In un tempo ragionevole, acquisirete una sufficiente familiarità con il vostro set di istruzioni e con i trucchi della programmazione, tanto da essere in grado di utilizzare l'esperienza formatasi nell'utilizzare un dato set di istruzioni, con altri set, indipendentemente dal tipo di microcomputer o minicomputer, o addirittura grande sistema di elaborazione dati.

Se siete interessati ai linguaggi alto livello, non avrete molto da aspettare. In aggiunta al package MITS BASIC, è ora disponibile un package BASIC 8080 del Livermore Laboratory ed un compilatore 8080 FORTRAN (Control Logic, Inc.). Il software Livermore è disponibile tramite l'Intel User's Group, «Insite».

Occorre comunque notare che in ogni caso dovrete imparare a programmare in assembler.

Semplici programmi e routine possono essere scritte facilmente e rapidamente sia in assembler che in linguaggio alto livello; tali programmi sono poi eseguiti più rapidamente, richiedendo meno memoria e, probabilmente, sono più facili da capire.

Del resto è fondamentale conoscere l'assembler, anche per capire tutti quei programmi, già esistenti e utilizzabili, scritti apposta in assembler. Infine, per essere in grado di effettuare delle valide analisi dei set di istruzioni di vari microprocessori, occorre sapere l'assembler. I programmi di cui avrete bisogno vi costeranno cari; sia che li facciate da voi, sia che li commissionate ad altri. Quindi se potete adottare altri programmi già esistenti alle vostre applicazioni: fatelo!... potrete realizzare un interessante risparmio.

FONTI DI INFORMAZIONE SOFTWARE PER L'8080

Vi forniamo un elenco di alcune ottime fonti di informazioni per il software 8080/8080A:

1. Intel Corporation, *Intel 8080 Microcomputer Systems User's Manual,* Intel Corporation, 3605 Bowers Avenue, Santa Clara, California 95051,

   Il Capitolo 4 fornisce un sommario del set di istruzioni dell'8080/8080A. Per ciascun tipo di istruzione, sono dati i cicli macchina richiesti dall'esecuzione, anche se vi sono due possibili tempi di esecuzione. Sono indicati i metodi di indirizzamento ed i flag modificati in seguito all'esecuzione delle singole istruzioni. Altri capitoli descrivono le funzioni di un computer, la CPU 8080, le tecniche di interfacciamento, la famiglia di chip di supporto 8080. Se dovrete fare del serio lavoro di progettazione vi serve questo manuale.

2. Intel Corporation, *Intel 8080 Assembly Language Programming Manual, Intel Corporation,* 3605 Bowers Avenue, Santa Clara California 95051.

   Un eccellente manuale che spiega argomenti come: program counter, stack pointer, rappresentazione del programma in memoria, indirizzamento di memoria, bit di condizione, linguaggio assembler e l'intero set 8080. Si discute anche l'uso delle MACRO, molto importanti nell'uso dell'assembler. Questo manuale vi è necessario, se dovete programmare con il cross assembler della Intel, o se dovete leggere programmi assemblati in modo cross. Molti programmi della libreria Intel possono essere compresi con l'aiuto di questo manuale.

3. NEC Microcomputers, Inc., *The μCOM-8 Software Manual,* NEC Microcomputers. Inc., 5 Militia Drive, Lexington, Massachusetts 02173,

   Un ottimo manuale, che tratta i seguenti problemi software:
   - Un semplice dispositivo di sense
   - Un counter gated
   - Un controller per un motore, programmato in tempo reale
   - Un branch ad N vie
   - Una subroutine di interrupt
   - Una subroutine di I/O per una TTY a 10 Hz
   - Una subroutine per somma o sottrazione a 16 digit

- Spostamento dati in memoria
- Macroprogrammazione e assemblaggio condizionato

Le singole istruzioni dell'8080 sono illustrate in modo eccellente. Sono presentati, per ogni problema, dei dettagliati flowchart. Per il lettore con qualche esperienza nell'assembler 8080, questo manuale sarà fonte di molte e utili tecniche di programmazione.

4. Intel Corporation, *Intel 8-bit User's Program Library, Intel Corporation,* User's Library, Microcomputer Systems, 3065 Bowers Avenue, Santa Clara, California 95051.

I programmi presentati alla User's Library devono essere accompagnati dal modulo della *Microcomputer User's Library Submittal Form.* Copie di questo modulo possono essere ordinate al Software Marketing Group della Intel. Questo modulo è usato dal Manager della User's Library per la preparazione del catalogo e degli aggiornamenti, e la descrizione della «Function» è usata nella preparazione dell'indice del catalogo che è inviato ai membri (si diventa membri presentando un programma che viene accettato oppure pagando una quota di 100 dollari). Questa Form è anche usata per ogni programma contenuto nella libreria. Sul retro della Library Submittal Form vi sono dettagliate istruzioni, che devono essere seguite nella preparazione di programmi da presentare.
Questi standard di documentazione sono mantenuti per assicurare l'utilizzo di ciascun programma della libreria da parte di ogni membro. Leggete attentamente i punti 2, 3 e 4 delle istruzioni contenute nella Library Submittal Form. *Il programma non può essere il duplicato di un programma già presente nella libreria.* Il programma deve essere senza errori e deve essere scritto nel linguaggio standard Intel (4004, 4040, 8008, 8080, PL/M). Occorre fornire un listing sorgente ed un nastro perforato. Il package originale della User's Library ebbe un primo aggiornamento nel dicembre del 1975, un secondo nel settembre 1976, seguito da aggiornamenti ogni due mesi. Nel settembre 1976, vi erano 200 programmi nella libreria. E' possibile avere i nastri perforati ad un costo molto basso, di semplice gestione della preparazione e spedizione dei nastri stessi. L'amministratore della User's Library è Ms. Marianne Vilas. Notate bene che tutti i programmi possono essere modificati e tagliati su misura per le vostre specifiche applicazioni Nel 1975, la Intel ha sponsorizzato un User's Library Contest di 22 settimane, al fine di ampliare la libreria stessa. Vi elenchiamo alcuni dei programmi che potete trovare nella libreria.

DATA ARRAY MOVE (8080). Un array contiguo di dati può essere rilocato in memoria, indipendentemente dall'ordine di grandezza della rilocazione richiesta. Le locazioni sorgente e destinazione dell'array possono sovrapporsi. La dimensione massima dell'array è 2^{16} byte.

PAPER TAPE LABELER (8080). Accetta da TTY dei caratteri ASCII e perfora il corrispondente carattere alfanumerico sul nastro.

TEXT STORAGE PROGRAM (8080). Permette la memorizzazione di un tasto in memoria, utilizzando una lettera dell'alfabeto come pointer. Dopo la memorizzazione del messaggio, questo può essere ritrovato semplicemente premendo un tasto della TTY. Si possono memorizzare fino ad un massimo di 32 messaggi.

CLOCK SUBROUTINE (8080). Mantiene il tempo del giorno, con aggiornamento decimale in BCD, delle ore, minuti e secondi. Può essere richiamato da hardware esterno ogni secondo, da un'interrupt esterno. Il tempo è memorizzato in 3 byte di memoria, secondo le 24 ore, oppure, opzionalmente, secondo un sistema basato sulle 12 ore.

TIMESHARING COMMUNICATIONS (8080). Per comunicare con un com-

puter a media o larga scala come utente esterno in un sistema timesharing.

IBM SELECTRIC OUTPUT PROGRAM (8080). Permette l'utilizzo della Selectric 731 IBM come dispositivo di uscita.

8080 IDLE ANALYZER FOR APPROXIMATING CPU UTILIZATION (8080). Visualizza il tempo che l'8080 spenderebbe in un «idle loop». Quando il RUN time è confrontato con l'idle time, si può calcolare la percentuale di utilizzo della CPU.

INTERRUPT SERVICE ROUTINE (8080). Gestisce gli interrupt multilevel, salvando i registri e i flag, ponendo in uscita lo stato dell'interrupt corrente verso un latch esterno.

8080 DIS-ASSEMBLER (8080 PL/M). Questo programma accetta un nastro esadecimale e genera un programma in linguaggio assembler simbolico.

MEMORY DIAGNOSTIC PROGRAM (8080). Scrive dei byte di test in qualsiasi range di memoria ed effettua una comparazione tra i bit scritti e quelli riletti. Viene stampato un messaggio di errore quando si scopre una locazione di memoria difettosa.

MATH (8080). Alcune routine sia per aritmetica floating-point che a virgola fissa insieme a un programma di dimostrazione che esegue valutazioni algebriche (da sinistra a destra senza alcuna precedenza dell'operatore) e che consente illimitate parentesi di nesting.

ELEMENTARY FUNCTION PACKAGE (8080). Effettua i seguenti valori floating-point con precisione su cinque digit decimali: radice quadrata, logaritmo, funzione ensponenziale, seno, coseno, arcotangente, seno e coseno iperbolico. Somma, sottrae, moltiplica e divide con precisione 7 digit decimali floating-point. (NOTA: Gli autori hanno avuto modo di utilizzare il programma che è risultato veramente ottimo. L'intero programma richiede circa 2500 byte di memoria).

8080 FLOATING-POINT PACKAGE WITH BCD CONVERSION ROUTINE (8080). Effettua somma floating-point, sottrazione, moltiplicazione, divisione, fixing, floating, negazione e conversione da floating-point BCD con esponente.

8080 LEAST-SQUARES QUADRATIC FITTING ROUTINE (8080). Gestione di matrici fino a un massimo di 256 coppie floating-point X, Y, verso una funzione del tipo $aX^2 + bX + C = Y$.

N-BYTE BINARY MULTIPLICATION AND LEADING ZERO BLANKING (8080). Il programma esegue una moltiplicazione binaria di due numeri con risultato di 255 byte come massimo.

8080 CROSS COMPILER ON THE PDP-11 (8080). Accetta un formato PDP-11 e produce un listing completamente codificato, una tabella dei simboli e un nastro perforato, utilizzabile per un loader standard.

PAGE LISTING PROGRAM (8080). Mette a disposizione delle facilitazioni per informazioni di listing in un formato numerato ed impaginato

SOURCE PAPER TAPE TO MAGNETIC CASSETTE (8080). Ricopia un nastro perforato in cassetta magnetica. Lo statement END deve essere seguito da CR (ritorno in carrello).

NATURAL LOGARITHM (8080). Calcola il logaritmo naturale dei numeri da 1 a 65.535.

BCD MULTIPLICATION (8080). Moltiplica fino a 6 digit BCD per 4 digit BCD, fornendo un risultato su 10 digit BCD. Tutti i numeri sono senza segno.

DOUBLE-PRECISION MULTIPLE (8080 PL/M). Moltiplica due numeri a 16 digit fornendo i 16 bit più significativi. La capacità intrinseca di moltiplicazione del PL/M è usata per le moltiplicazioni byte per byte.

SUBROUTINE OG. Questa subroutine fornisce il log di qualsiasi base intera di qualsiasi numero positivo floating-point.

5. Scelbi Computer Consulting, Inc., 1322 Rear Post Road, Milford, Connecticut 06460.

E' disponibile il seguente software:
. *Machine Language Programming for the 8080 (and similar microcomputers)*
. *An 8080 Assembler Program*
. *An 8080 Editor Program*
. *8080 Monitor Routines*
. *SCELBAL. Scientific Elementary Basic Language for 8008/8080, Systems*
. *SCELBI's First Book of Computer Games for the 8008/8080, SCELBI's GALAXY GAME for the 8008/8080.*

Potete trarre molte interessanti tecniche di programmazione da quanto esposto nel testo. Non sono disponibili nastri perforati.

6. Zilog Corporation, *Z80-CPU Technical Manual,* Zilog, Inc., 170 State Street, Los Altos, California 94022.

Non otterrete molti suggerimenti da questo manuale, ma è comunque molto interessante effettuare un confronto tra lo Z-80 e l'8080A in termini di set di istruzioni.

7. *Byte,* Byte Pubblications, Inc., 70 Main Street, Peterboroungh, New Hampshire 03458.

I singoli articoli variano di qualità; in ogni caso sono reperibili interessanti suggerimenti e tecniche di programmazione. La rivista è una delle più lette da chi si occupa di hobby computer.

8. National Semiconductor Corp., *PACE Logic Designer's Guide to Programmed Equivalents to TTL Functions,* National Semiconductor Corporation, 2900 Semiconductor Drive, Santa Clara, Californa 95051.

Questo testo è redatto per il microprocessore PACE, un 16 bit. E' molto ben fatto per quanto concerne la sostituzione dell'hardware con il software. Con una minima conoscenza del set di istruzioni del PACE, potreste essere in grado di convertire i programmi in linguaggio 8080.

9. *Kilobaud,* Kilobaud, Peterborough, New Hampshire 03458.

Nella stessa categoria di BYTE, cioè rivista orientata all'hobby computer con interessanti suggerimenti sulle tecniche di programmazione.

10. Adam Osborne and Associates, Inc., P.O. Box 2036, Berkeley, California 94702.

Sono disponibili i seguenti libri sull'8080:
. *An introduction to Microcomputers. Volume I. Basic Concepts*
. *An Introduction to Microcomputers. Volume II. Some Real Products*
. *8080 Programming for Logic Design*

Tutti questi testi sono fondamentali per un serio utente dell'8080. Questi libri saranno editi in lingua italiana probabilmente distribuiti in Italia entro il 1978 dalla Jackson Italiana Editrice.

11. W. J. Weller, A. V. Shatzel, and H. Y. Nice, *Practical Microcomputer Programming. The Intel 8080,* Northern Technology Books, P. O. Box 62, Evanston, Illinois 60204.

Riteniamo questo libro utile al fine di sviluppare una esperienza software.

SOMMARI DEL SET DI ISTRUZIONI DELL'8080

Alcuni sommari del set di istruzioni dell'8080, sono disponibili dalle seguenti fonti:

1. Intel Corporation, *Intel 8080 Assembly Language Reference Card,* Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051.

   Fornisce l'elenco esadecimale del set di istruzioni dell'8080 e il relativo elenco funzionale.

2. Tychon, Inc., *8080 Octal/Hexadecimal Code Cards,* Tychon, Inc., P. O. Box 242, Blacksburg, Virginia 24060.

   Interessante sistema «a regolo», col quale è possibile avere immediatamente la traduzione esadecimale delle possibili combinazioni delle istruzioni dell'8080. Può essere direttamente acquistato alla Microlem di Milano.

3. Martin Research, *8080 Instruction Set,* 3336 Commercial Ave., North-Brook, Illinois 60062.

   Suddivide il set di istruzioni 8080 per funzione.

4. R. Baker, *Byte,* p. 84 (February 1976).

   Listing compatto del codice ottale del set di istruzioni dell'8080.

5. P. R. Rony, D. G. Larsen, and J. A. Titus, *The 8080A Bugbook: Microcomputer Interfacing and Programming,* Howard W. Sams & Co., Inc., 4300 West 62nd St., Indianapolis, Indiana 46268.

   Il set di istruzioni dell'8080 è dato come listing dei gruppi di istruzioni, listing alfabetico dei codici menmonici, listing numerico ottale/esadecimale.

DESCRIZIONE DELLE SINGOLE ISTRUZIONI DELL'8080

Nelle pagine che seguono forniamo una descrizione dettagliata del set dell'8080. Utilizziamo del materiale tratto sia dall'*Intel 8080 Microcomputer Systems User's Manual,* che dal *μCOM-8 Software Manual,* per concessione, rispettivamente della Intel Corp. e della NEC Microcomputers Inc.

Le istruzioni sono, dalla Intel, suddivise nei seguenti gruppi:

- GRUPPO TRASFERIMENTO DATI: spostamento dati tra registri o tra registri e memoria.
- GRUPPO ARITMETICO: somma, sottrazione, incremento, decremento di dati sia nei registri che in locazioni di memoria.
- GRUPPO LOGICO: AND. OR, OR esclusivo, comparazione, rotazione, complementazione di dati in registri di memoria.
- GRUPPO DI BRANCH: Istruzioni di salti condizionati e incondizionati, istruzioni di richiamo a subroutine e ritorno da subroutine.
- STACK, I/O E GRUPPO CONTROLLO MACCHINA: include le istruzioni di I/O, di gestione stack o flag interni di controllo.

SET DI ISTRUZIONI

Un computer, per quanto "raffinato", può fare solo quello che gli viene "detto" di fare. Si dice al computer cosa fare attraverso una serie di istruzioni codificate a cui ci si riferisce come ad un **Programma.** Il regno del programmatore è costituito dal **Software,** al contrario dell'**Hardware** che è compreso negli attuali computer. Il software di un computer fa riferimento a tutti i programmi che sono stati scritti per quel computer.

Nella progettazione di un computer, il tecnico fornisce alla CPU la capacità di eseguire un particolare set di operazioni. La CPU è realizzata in modo tale che viene eseguita un'operazione specifica quando la logica di controllo della CPU stessa decodifica una particolare istruzione. Di conseguenza, le operazioni che possono essere eseguite dalla CPU definiscono il **Set di Istruzioni** del computer.

Ogni istruzione permette al programmatore di iniziare l'esecuzione di una specifica operazione. Tutti i computer implementano determinate operazioni aritmetiche nel loro set di istruzioni, come un'istruzione di somma del contenuto di due registri. Spesso sono comprese nel set di istruzioni operazione logiche (ad esempio OR del contenuto di due registri) e istruzioni operative su un registro (ad esempio incremento di un registro). Il set di istruzioni di un computer avrà anche istruzioni che spostano i dati fra i registri, fra un registro e la memoria e fra un registro e un dispositivo di I/O. La maggior parte dei set di istruzioni fornisce anche le **Istruzioni Condizionali.** Un'istruzione condizionale specifica un'operazione che va eseguita solo se si sono verificate determinate condizioni; per esempio, saltare ad una particolare istruzione se il risultato dell'ultima operazione era zero. Le istruzioni condizionali forniscono un programma in grado di "prendere decisioni".

Organizzando in modo logico una sequenza di istruzioni all'interno di un programma coerente, il programmatore può "dire" al computer di eseguire una funzione specifica e molto utile.

Il computer, comunque, può eseguire solo programmi le cui istruzioni sono in forma di codice binario (ad es. una serie di 1 e 0), che viene chiamato **Codice Macchina.** Dato che sarebbe estremamente scomodo programmare in codice macchina, sono stati sviluppati i linguaggi di programmazione. Sono disponibili programmi che convertono le istruzioni scritte in linguaggio di programmazione, in codice macchina, che può venire interpretato dal processore.

Un tipo di linguaggio per programmare è il **Linguaggio Assembler.** Ad ognuna delle istruzioni del computer viene assegnato un solo codice mnemonico del linguaggio assembler. Il programmatore può scrivere un programma (chiamato **Programma Sorgente)** usando questi codici mnemonici e determinati operandi; il programma sorgente viene poi convertito in istruzioni macchina (chiamate il **Codice Oggetto).** Ogni istruzione del linguaggio assembler viene poi convertita in un'istruzione di codice macchina (1 o più byte) da un programma **Assembler.** I linguaggi assembler dipendono solitamente dalla macchina (ed essi di solito sono in grado di funzionare solo su di un tipo di computer).

IL SET DI ISTRUZIONI DEL MICROPROCESSORE 8080

Il set di istruzioni dell'8080 comprende cinque diversi tipi di istruzioni:

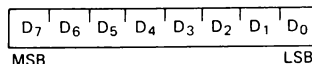
- **Gruppo Trasferimento Dati** - sposta i dati fra un registro e l'altro o fra la memoria e i registri
- **Gruppo Aritmetico** - addiziona, sottrae, incrementa o decrementa i dati nei registri o nella memoria
- **Gruppo Logico** - AND, OR, OR ESCLUSIVO, confronta, fa ruotare o complementa i dati nei registri o nella memoria
- **Gruppo Branch** - istruzioni di salto condizionato e incondizionato, istruzioni di chiamata della subroutine e istruzioni di rientro
- **Gruppo Stack, I/O e Controllo Macchina** - comprende le istruzioni I/O, nonchè le istruzioni per mantenere lo stack e i flag di controllo interni.

Formato delle Istruzioni e dei Dati:

La memoria per l'8080 è organizzata in quantità di 8 bit, chiamati byte. Ogni byte ha un solo indirizzo binario a 16 bit che corrisponde alla sua posizione sequenziale in memoria.

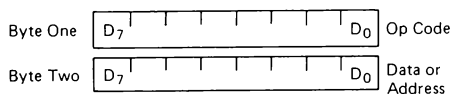
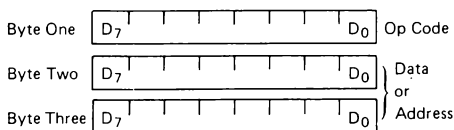
L'8080 può indirizzare direttamente fino a 65.536 byte di memoria, che possono essere costituiti sia da elementi di memoria di sola lettura (ROM) che da elementi di memoria ad accesso casuale (RAM) (memoria di lettura/scrittura).

I dati nell'8080 vengono memorizzati sotto forma di numeri interni binari a 8 bit:

PAROLA DATI

Quando un registro o una parola dati contiene un numero binario, è necessario stabilire l'ordine nel quale sono scritti i bit del numero. Nell'8080, ci si riferisce al BIT 0 come al **bit meno significativo (LSB)** e al BIT 7 (di un numero di 8 bit) come al **bit più significativo (MSB).**

Le istruzioni di programma dell'8080 possono essere una, due o tre byte in lunghezza. Le istruzioni a più byte devono essere memorizzate in successive locazioni di memoria; l'indirizzo del primo byte viene sempre usato come indirizzo delle istruzioni. L'esatto formato dell'istruzione dipenderà dalla particolare operazione che deve essere eseguita.

Istruzioni ad un byte

Istruzioni a due byte

Istruzioni a tre byte

Modi di Indirizzamento

Spesso i dati sui quali bisogna operare sono immagazzinati in memoria. Quando si usano dati numerici a più byte, i dati, come le istruzioni, vengono memorizzati in posizioni di memoria successive, con il byte meno significativo al primo posto, seguito via via dai byte più significativi. L'8080 ha quattro modi diversi di indirizzare i dati caricati in memoria o nei registri:

- Diretto - I byte 2 e 3 dell'istruzione contengono l'esatto indirizzo di memoria dei dati in questione (i bit d'indirizzo di ordine inferiore sono nel byte 2, i bit di ordine più elevato sono nel byte 3).
- Registro - L'istruzione specifica il registro o la coppia di registri in cui i dati sono posizionati.
- Registro Indiretto - L'istruzione specifica una coppia di registri che contiene l'indirizzo di memoria in cui i dati sono posizionati (i bit di ordine più elevato sono nel primo dei due indirizzi, i bit di ordine inferiore nel secondo).
- Immediato- L'istruzione contiene i dati stessi. Questi sono in quantità o di 8 o di 16 bit (il byte meno significativo per primo, il byte più significativo per secondo).

A meno che non venga diretta da un'istruzione d'interruzione o di salto, l'esecuzione delle istruzioni procede attraverso posizioni di memoria che aumentano sequezialmente. Un'istruzione di salto può specificare l'indirizzo dell'istruzione successiva che deve essere eseguita in uno dei due modi seguenti:

- Diretto - L'istruzione di salto contiene l'indirizzo dell'istruzione seguente che va eseguita (Eccetto che per l'istruzione 'RST' il byte 2 contiene l'indirizzo di ordine inferiore e il byte 3 l'indirizzo di ordine più elevato).
- Registro Indiretto - L'istruzione di salto indica una coppia di registri che contiene l'indirizzo dell'istruzione che va eseguita successivamente. (I bit d'indirizzo di ordine più elevato sono nel primo dei due registri, i bit di ordine inferiore nel secondo).

L'istruzione RST è una speciale istruzione di richiamo ad un byte (usata di solito durante le sequenze d'interruzione). RST comprende un campo a 3 bit; il controllo del programma viene trasferito all'istruzione il cui indirizzo è otto volte il contenuto di questo campo a tre bit.

Flag di Condizione (Condition Flags):

Ci sono cinque flag di condizione collegati con l'esecuzione delle istruzioni sull'8080. Essi sono Zero, Sign, Parity, Carry, e Auxiliary Carry, ed ognuno di loro è rappresentato da un registro ad 1 bit nella CPU. Un flag viene "settato" forzando il bit verso 1, "resettato" forzando il bit verso 0.

Salvo diversa indicazione, quando un'istruzione coivolge un flag lo fa nel modo seguente:

Zero: Se il risultato di un'istruzione ha il valore 0, questo flag è settato; altrimenti viene resettato.

Sign: Se il bit più significativo del risultato dell'operazione ha il valore 1, questo flag è settato; altrimenti viene resettato.

Parity: Se la somma modulo 2 dei bit del risultato dell'operazione è 0, (ad es. se il risultato ha parità even), questo flag viene settato; altrimenti viene resettato (ad es. se il risultato ha parità odd).

Carry: Se l'istruzione determina un riporto (da addizione), o un riporto negativo (da una sottrazione o da una comparazione) fuori dal bit più significativo, questo flag viene settato; altrimenti viene resettato.

Auxiliary Carry:	Se l'istruzione ha dato luogo ad un riporto dal bit 3 al bit 4 del valore risultante, il riporto ausiliario viene settato; altrimenti, è resettato. Questo flag riguarda le addizioni, le sottrazioni, gli incrementi, i decrementi, le comparazioni e le operazioni logiche, ma viene usato principalmente con le addizioni e gli incrementi che precedono un'istruzione DAA (Decimal Adjust Accumulator)

Simboli ed Abbreviazioni

I simboli e le abbreviazioni che seguono vengono usati nella descrizione successiva delle istruzioni dell'8080:

SIMBOLI	**SIGNIFICATO**
accumulatore	registro A
addr	quantità d'indirizzo a 16 bit
data	quantità di dati a 8 bit
data 16	quantità di dati a 16 bit
byte 2	il secondo byte dell'istruzione
byte 3	il terzo byte dell'istruzione
port	indirizzo a 8 bit di un dispositivo I/O
r, r1, r2	uno dei registri A,B,C,D,E,H,L
DDD,SSS	la configurazione di bit che designa uno dei registri A,B,C,D,E,H,L (DDD = destinazione, SSS = sorgente):

DDD o SSS	**NOME DEL REGISTRO**
111	A
000	B
001	C
010	D
011	E
100	H
101	L

rp	una delle coppie di registri: B rappresenta la coppia B,C dove B è il registro di ordine più elevato e C il registro di ordine inferiore; D rappresenta la coppia D,E dove D è il registro di ordine più elevato ed E il registro di ordine inferiore; H rappresenta la coppia H,L dove H è il registro di ordine più elevato e ed L il registro di ordine inferiore SP rappresenta il registro stack pointer a 16 bit
RP	La configurazione di bit che designa una delle coppie di registri B,D,H,SP:

RP	**COPPIA DI REGISTRI**
00	B-C
01	D-E
10	H-L
11	SP

rh	Il primo registro (ordine più elevato) di una coppia di registri designata.
rl	Il secondo registro (ordine inferiore) di una coppia di registri designata.
PC	Registro contatore di programma a 16 bit (PCH e PCL vengono usati per riferirsi rispettivamente agli 8 bit di ordine più elevato e a quelli di ordine inferiore.
SP	Registro stack pointer a 16 bit (SPH e SPL vengono usati per riferirsi rispettivamente agli 8 bit di ordine più elevato e a quelli di ordine inferiore).
rm	Bit m del registro r (i bit vanno dal numero 7 allo 0 da sinistra verso destra).
Z, S, P, CY, AC	I flag di condizione: Zero, Sign, Parity, Carry, e Auxiliary Carry, rispettivamente.
( )	Il contenuto della locazione di memoria o dei registri chiuso in parentesi
←	"Viene trasferito a"
∧	AND logico
⊻	OR esclusivo
V	OR inclusivo
+	Addizione
−	Sottrazione di complemento di due
*	Moltiplicazione
↔	"Viene scambiato con"
———	Il complemento ad uno (es. (A))
n	Numeri da 0 a 7
NNN	La rappresentazione binaria da 000 a 111

Descrizione del Format:

Le pagine seguenti presentano una descrizione dettagliata del set di istruzioni dell'8080. Ogni istruzione è descritta nel modo seguente:

1. Il formato assembler MAC 80, che consiste del codice mnemonico di istruzione e dei campi operandi, è stampato in **NERETTO** sul lato sinistro della prima riga.
2. Il nome dell'istruzione è chiuso fra parentesi sul lato destro della prima riga.
3. La riga o le righe successive contengono una descrizione simbolica dell'operazione dell'istruzione.
4. Segue quindi una descrizione dell'operazione dell'istruzione.
5. La riga o le righe seguenti contengono le configurazioni e i campi binari che comprendono l'istruzione macchina.

6. Le ultime quattro righe contengono informazioni incidentali circa l'esecuzione dell'istruzione. Per primo è elencato il numero di stati e di cicli macchina richiesto per eseguire l'istruzione. Se l'istruzione ha due tempi di esecuzione possibili, come nel Salto Condizionato, saranno indicati tutti e due i tempi, separati da un trattino. Quindi, sono mostrati tutti i modi significativi di indirizzamento dei dati . L'ultima riga indica alcuni dei cinque flag che sono coinvolti nell'esecuzione dell'istruzione.

GRUPPO TRASFERIMENTO DATI

Questo gruppo di istruzioni trasferisce i dati da e verso i registri e la memoria. In questo gruppo i flag di condizione non vengono coinvolti da nessuna istruzione.

MoV r1, r2

MOV r1, r2 (Spostare il registro)

(r1) – (r2)
Il contenuto del registro r2 è spostato al registro r1.

0	1	D	D	D	S	S	S

Cicli: 1
Stati: 5
Indirizzamento: Registro
Flag: nessuno

L'istruzione MOV r1, r2 trasferisce i dati dal registro sorgente S specifico (r2) al registro destinazione D specifico (r1). La sorgente o la destinazione possono essere uno qualunque dei singoli registri B, C, D, H, o L, l'accumulatore A, e M (il contenuto dell'indirizzo di memoria specificato dalla coppia di registri H, L). Nel byte a tre cifre ottali, la prima cifra è sempre un 1. La seconda e la terza cifra ottale variano a seconda della sorgente e della destinazione.
L'istruzione ottale, 166, è un'istruzione di arresto piuttosto che un'istruzione MOV. Il contenuto del registro sorgente non cambia durante un'istruzione MOV; esso viene duplicato.

MOV r,M

L'istruzione MOV r,M trasferisce i dati da M (il contenuto dell'indirizzo di memoria specificato dalla coppia di registri H,L) al registro destinazione D specifico, che può essere uno qualunque dei singoli registri B, C, D, H, o L o l'accumulatore A. Il contenuto dell'indirzzo di memoria viene duplicato nel registro; il contenuto della memoria resta invariato.

MOV r,M (Spostare dalla memoria)
(r) ← (H) (L)
Il contenuto della locazione di memoria, il cui indirizzo è nei registri H ed L, viene spostato nel registro r.

Cicli: 2
Stati: 7
Indirizzamento: reg. indiretto
Flag: nessuno

MOV M,r

MOV M,r (Spostare verso la memoria)
((H) (L)) ← (r)
Il contenuto del registro r viene spostato verso la locazione di memoria il cui indirizzo è nei registri H e L.

Cicli: 2
Stati: 7
Indirizzamento: reg. indiretto
Flag: nessuno

L'istruzione MOV M,r trasferisce i dati dal registro sorgente S specificato, ad M (l'indirizzo di memoria specificato dalla coppia di registri H,L). Il registro sorgente può essere uno qualunque dei singoli registri B, C, D, E, H, o L o l'accumulatore A. I contenuti del registro vengono duplicati in memoria; i contenuti del registro restano invariati.

MVI r, data

MVI r, data (Spostare in modo immediato)
(r) ← (byte 2)
Il contenuto del byte 2 dell'istruzione viene spostato verso il registro r.

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: nessuno

L'istruzione MVI r, data trasferisce i dati dal secondo byte dell'istruzione a due byte al registro destinazione D specificato (r). Il termine *"immediato"* si riferisce al fatto che il byte di dati è contenuto all'interno dell'istruzione a più byte. Il registro destinazione specificato può essere uno dei singoli registri B, C, D, E, H o L, o l'accumulatore A, e M (i contenuti dell'indirizzo di memoria specificato dalla coppia di registri H,L). Quando la destinazione è M, avete l'istruzione MVI M, data, trattata qui sotto. I dati possono essere costituiti da qualunque numero binario a 8 bit fra 00000000 e 11111111.

MVI M, data

MVI M, data (Spostare in modo immediato verso la memoria)
(H) (L) ← (byte 2)
Il contenuto del byte 2 dell'istruzione viene spostato nella locazione di memoria il cui indirizzo è nei registri H ed L.

Cicli: 3
Stati: 10
Indirizzamento: immediato/reg. indiretto
Flag: nessuno

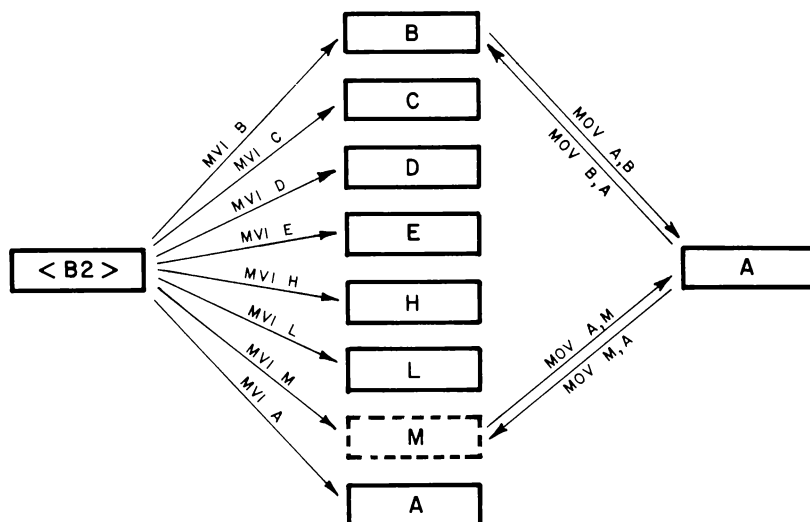
L'istruzione MVI M, data trasferisce i dati dal secondo byte dell'istruzione ad M (l'indirizzo di memoria specificato dalla coppia di registri H,L). I dati possono essere costituiti da qualunque numero binario a 8 bit fra 00000000 e 11111111.

LXI rp,data 16

LXI rp,data 16 (Caricare in modo immediato la coppia di registri)
(rh) ← (byte 3),
(rl) ← (byte 2)
Il byte 3 dell'istruzione viene spostato nel registro di ordine più elevato (rh) della coppia di registri rp. Il byte 2 dell'istruzione viene spostato nel registro di ordine inferiore (rl) della coppia di registri rp.

Cicli: 3
Stati: 10
Indirizzamento: immediato
Flag: nessuno

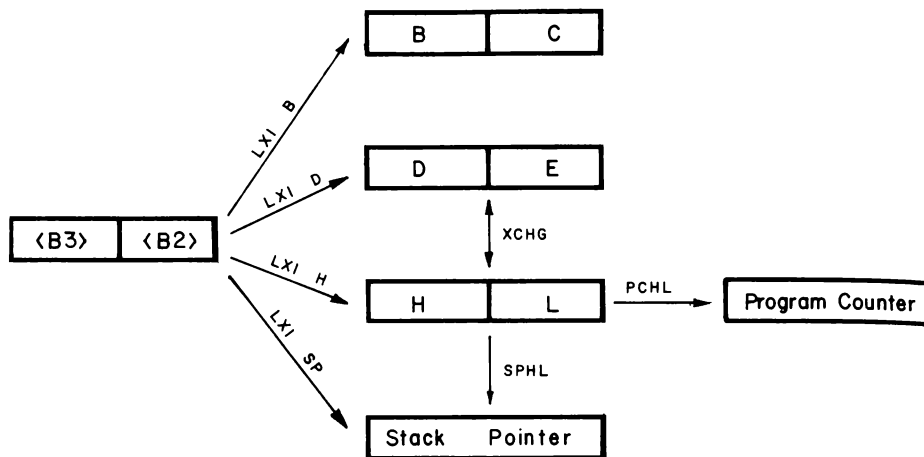
L'istruzione LXI rp, data fa sì che una quantità di dati a 16 bit contenuta nel secondo e nel terzo byte dell'istruzione venga caricata nella coppia di registri specificata da RP. RP può essere una qualunque delle coppie di registri HL, DE o BC o lo stack pointer, che sono rappresentati dai codici mnemonici H, D, B e SP, rispettivamente.
Il secondo byte dell'istruzione viene caricato nei registri LO - L, E, C, o negli 8 bit LO dello stack pointer; il terzo byte dell'istruzione è caricato nei registri HI - H, D, B o negli otto bit HI dello stack pointer. La parola dati a 16 bit può variare da 0000000000000000 a 1111111111111111, in notazione binaria.

Gli schemi seguenti illustrano alcune delle caratteristiche delle istruzioni MOV, MVI e LXI. Vengono mostrati solo due set delle istruzioni MOV r1, r2.

Da notare che LXI rp, data è equivalente a due istruzioni MVI r,data. Così,

LXI B
<B2>
<B3>

è equivalente a

MVI B
<B2> *(corrisponde a B3 nell'istruzione LXI B)*
MVI C
<B2> *(corrisponde a B2 nell'istruzione LXI B)*

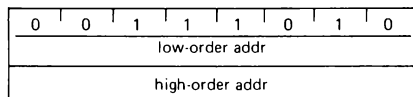
In un'istruzione a due byte, il secondo byte è indicato sempre come <B2>. Una sola istruzione LXI rp, data richiede 10 stati per essere eseguita, mentre due istruzioni MVI r,data richiedono in tutto un tempo di esecuzione di 14 stati. Perciò, usando l'istruzione LXI rp,data, risparmiate 4 stati del tempo di esecuzione. In molti casi, un tale risparmio non ha importanza.

STA addr

STA addr (Memorizzare direttamente l'accumulatore)
((byte 3) (byte 2)) ← (A)
Il contenuto dell'accumulatore viene spostato nella locazione di memoria il cui indirizzo è specificato nei byte 2 e 3 dell'istruzione.

Cicli: 4
Stati: 13
Indirizzamento: diretto
Flag: nessuno

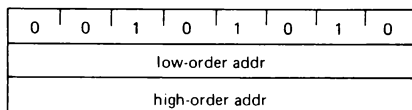
L'istruzione STA addr vi permette di memorizzare i contenuti dell'accumulatore direttamente nella locazione di memoria senza usare la coppia di registri H,L. L'indirizzo della locazione di memoria è specificato nel secondo e nel terzo byte dell'istruzione. Il byte d'indirizzo LO è il byte 2 e il byte d'indirizzo HI è il byte 3.

LDA addr

LDA addr (Caricare direttamente l'accumulatore)
(A) ← ((byte 3) (byte 2))
Il contenuto della locazione di memoria, il cui indirizzo è specificato nei byte 2 e 3 dell'istruzione, viene spostato al registro A.

Cicli: 4
Stati: 13
Indirizzamento: diretto
Flag: nessuno

L'istruzione LDA addr vi permette di caricare l'accumulatore con i contenuti della locazione di memoria specificata dai byte <B2> e <B3> nell'istruzione. Non avete bisogno di usare la coppia di registri H,L. Il byte d'indirizzo LO è <B2> e il byte d'indirizzo HI è <B3>.

LHLD addr

LHLD addr (Caricare direttamente H ed L)
(L) ← ((byte 3) (byte 2))
(H) ← ((byte 3) (byte 2) + 1)
Il contenuto della locazione di memoria, il cui indirizzo è specificato nei byte 2 e 3 dell'istruzione, viene spostato al registro L. Il contenuto della locazione di memoria all'indirizzo successivo viene spostato al registro H.

Cicli: 5
Stati: 16
Indirizzamento: diretto
Flag: nessuno

Questa istruzione è utile quando le locazioni di memoria contengono l'informazione d'indirizzo.

In questo caso LHLD addr fa sì che il registro H venga caricato con il byte di memoria indirizzato dai byte <B2> e <B3> nell'istruzione, es. addr. Il registro H viene caricato con il byte di memoria posizionato a addr + 1. Si esegue così un trasferimento a 16 bit di un indirizzo di memoria alla coppia di registri H,L. Una volta studiato XCHG, osserverete che la sezione di programma.

```
LHLD
<B2>
<B3>
XCHG
```

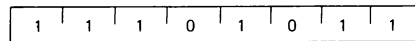
è funzionalmente equivalente a

```
LXI H
<B2>
<B3>
MOV E,M
INX H
MOV D,M
```

La prima sezione richiede 20 stati per l'esecuzione; la seconda sezione richiede 29 stati.

XCHG

XCHG (Scambiare H ed L con D e E)
(H) ←→ (D)
(L) ←→ (E)
I contenuti dei registri H ed L vengono scambiati con i contenuti dei registri D ed E.

1	1	1	0	1	0	1	1

Cicli: 1
Stati: 4
Indirizzamento: registro
Flag: nessuno

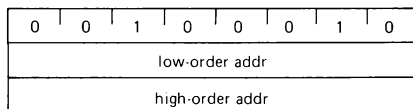
L'istruzione XCHG fa sì che i contenuti delle coppie di registri D,E e H,L vengano scambiati fra loro. Per essere precisi, si scambiano i contenuti dei registri D e H e dei registri E ed L. Questa istruzione vi permette di usare la coppia di registri H,L come un indirizzo di memoria, mentre un altro indirizzo di memoria viene conservato nella coppia di registri D,E. Potete modificare i contenuti della coppia di registri D,E senza cambiare la coppia di registri H,L. Per esempio, la coppia di registri H,L può specificare una posizione di memoria che usate per modificare la coppia di registri D,E. Due istruzioni XCHG in sequenza,

```
XCHG
XCHG
```

equivalgono ad una "no operation."

SHLD addr

SHLD addr (Memorizzare direttamente H ed L)
((byte 3) (byte 2)) ← (L)
((byte 3) (byte 2) + 1) ← (H)
Il contenuto del registro L viene spostato nella locazione di memoria il cui indirizzo è specificato nei byte 2 e 3. Il contenuto del registro H viene spostato nella posizione di memoria successiva.

0	0	1	0	0	0	1	0
low-order addr							
high-order addr							

Cicli: 5
Stati: 16
Indirizzamento: diretto
Flag: nessuno

L'istruzione SHLD addr fa sì che i contenuti del registro L vengano memorizzati nella locazione di memoria data dai byte <B2> e <B3> nell'istruzione, es. addr. I contenuti del registro H vengono memorizzati nella locazione di memoria, addr + 1. In altre parole, si esegue un trasferimento a 16 bit di 2 byte d'indirizzo della coppia di registri H,L verso le due locazioni di memoria successive, addr + 1. Questa istruzione è utile nel creare un gruppo di locazioni di memoria che contengono informazioni sull'indirizzo piuttosto che dati. Come con la maggior parte delle istruzioni dell'8080A, il byte <B2> è il byte d'indirizzo LO e il byte <B3> è il byte d'indirizzo HI addr.

La sezione di programma

```
XCHG
SHLD
<B2>
<B3>
```

è equivalente alla sezione di programma,

```
LXI H
<B2>
<B3>
MOV M,E
INX H
MOV M,D
```

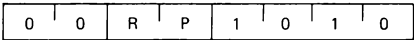
LDAX rp

L'istruzione LDAX rp vi permette di caricare l'accumulatore con i contenuti della locazione di memoria indirizzata da una coppia di registri piuttosto che dalla coppia di registri H,L. Così, con LDAX B, usate la coppia di registri B,C per fornire l'indirizzo di memoria a 16 bit; con LDAX D, usate la coppia di registri D,E per fornire l'indirizzo. La sezione di programma

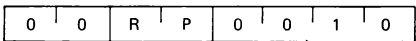
LXI D
<B2>
<B3>
LDAX D

carica l'accumulatore come

LXI H
<B2>
<B3>
MOV A,M

LDAX rp (Caricare indirettamente l'accumulatore)
(A) ← ((rp))
Il contenuto della locazione di memoria, il cui indirizzo è nella coppia di registri rp, viene spostato nel registro A. Nota: solo le coppie di registri rp=B (registri B e C) o rp = D (registri D ed E) possono essere specificate.

Cicli: 2
Stati: 7
Indirizzamento: reg. indiretto
flag: nessuno

STAX rp

STAX rp (Memorizzare indirettamente l'accumulatore)
((rp)) ← (A)
Il contenuto del registro A viene spostato nella locazione di memoria il cui indirizzo è nella coppia di registri rp. Nota: solo le coppie di registri rp=B (registri B e C) o rp=D (registri D ed E) possono essere specificate.

Cicli: 2
Stati: 7
Indirizzamento: reg. indiretto
Flag: nessuno

L'istruzione STAX rp vi permette di memorizzare i contenuti dell'accumulatore nella locazione di memoria indirizzata sia dalla coppia di registri B,D che dalla coppia di registri D,E. La sezione di programma

LXI B
<B2>
<B3>
STAX B

attua lo store dell'accumulatore come

LXI H
<B2>
<B3>
MOV M,A

Il significato delle istruzioni STAX rp e LDAX rp è che avete tre indirizzi di memoria a 16 bit indipendenti memorizzati nei registri universali all'interno del microprocessore 8080A. Sono disponibili sufficienti istruzioni da permettervi di usare tutti e tre gli indirizzi.

I flag di condizione non sono coinvolti da nessuna delle istruzioni della lista seguente:

MOV r1,r2
MOV r,M
MOV M,r
MVI r,data
MVI M,data
LXI rp, data 16
STA addr
LDA addr
XCHG
LHLD addr
SHLD addr
LDAX rp
STAX rp

Queste istruzioni costituiscono il gruppo trasferimento dati nel microprocessore 8080A.

GRUPPO ARITMETICO

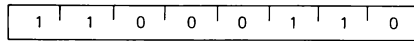
Questo gruppo di istruzioni esegue operazioni aritmetiche su dati nei registri e nella memoria. *Salvo diversa indicazione, tutte le istruzioni di questo gruppo coinvolgono i flag Zero, Sign, Parity, Carry e Auxiliary Carry secondo le regole standard.* Tutte le operazioni di sottrazione sono eseguite per mezzo dell'aritmetica complemento a due e posizionano il riporto ad uno per indicare un riporto negativo, e lo azzerano se non vi è nessun riporto negativo.

ADD r

ADD r (Addizionare il registro)
(A) ← (A) + (r)
Il contenuto del registro r è addizionato al contenuto dell'accumulatore. Il risultato è posto nell'accumulatore.

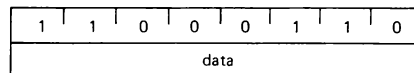
L'istruzione ADD r fa sì che i contenuti del registro sorgente S vengano addizionati ai contenuti dell'accumulatore. Il registro sorgente può essere uno qualunque dei registri universali B,C,D,E,H,L, l'accumulatore A, o M (i contenuti della memoria indirizzati dalla coppia di registri H,L). L'istruzione ADD M è descritta qui sotto. L'istruzione coinvolge tutti e quattro i flag testabili: Carry, Parity, Zero, e Sign. È coinvolto anche l'Auxiliary Carry.

ADD M

ADD M (Addizionare la memoria)
(A) ← (A) + ((H) (L))
Il contenuto della locazione di memoria il cui indirizzo è contenuto nei registri H ed L è addizionato al contenuto dell'accumulatore. Il risultato si trova nell'accumulatore.

Cicli: 2
Stati: 7
Indirizzamento: reg. indiretto
Flag: Z,S,P,CY,AC

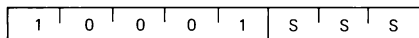
L'istruzione ADD M fa sì che i contenuti della locazione di memoria M, che è indirizzata dalla coppia di registri H,L, vengano addizionati ai contenuti dell'accumulatore. I contenuti della memoria restano invariati dopo l'addizione. L'istruzione coinvolge tutti e cinque i flag ed ha due cicli macchina.

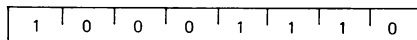
ADI data

ADI data (Addizionare immediatamente)
(A) ← (A) + (byte 2)
Il contenuto del secondo byte dell'istruzione è addizionato al contenuto dell'accumulatore. Il risultato si trova nell'accumulatore.

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: Z,S,P,CY,AC

L'istruzione ADI data fa sì che i dati presenti nel secondo byte dell'istruzione vengano addizionati ai contenuti dell'accumulatore. L'istruzione coinvolge tutti e cinque i flag.

ADC r e ADC M

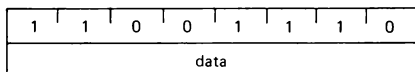
ADC r (Addizionare il registro con riporto)
(A) ← (A) + (r) + (CY)
Il contenuto del registro r e il contenuto del bit di riporto sono addizionati al contenuto dell'accumulatore. Il risultato si trova nell'accumulatore.

Cicli: 1
Stati: 4
Indirizzamento: registro
Flag: Z,S,P,CY,AC

ADC M (Addizionare la memoria con riporto)
(A) ← (A) + ((H) (L)) + (CY)
Il contenuto della locazione di memoria il cui indirizzo è contenuto nei registri H ed L e il contenuto del flag CY sono addizionati all'accumulatore. Il risultato si trova nell'accumulatore.

Cicli: 2
Stati: 7
indirizzamento: registro indiretto
Flag: Z,S,P,CY,AC

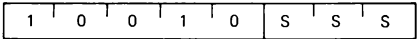
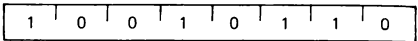
Citiamo il *"μCOM-8 Software Manual"*: "Allo scopo di eseguire operazioni di addizione e sottrazione, occorrono delle speciali istruzioni aritmetiche. L'aritmetica a più cifre richiede che due elementi vengano visualizzati e conservati da qualche parte. Questi due elementi sono la somma delle cifre come sono state addizionate e la presenza o l'assenza di un bit di riporto. Quando viene prodotto un bit di riporto, esso deve essere addizionato alla somma delle cifre seguenti. In modo analogo, con le operazioni di sottrazione, l'esistenza di un riporto negativo dev'essere individuata, in modo che esso possa venire trattenuto dalla differenza delle cifre seguenti. Le istruzioni Addizione con Riporto e Sottrazione con Riporto Negativo fanno in modo che vengano visualizzati e conservati i bit di riporto, facendo addizioni e sottrazioni a più cifre in modo molto chiaro. ADC r, ADC M e ACI data sono le istruzioni "Addizione con Riporto". ADC r fa sì che i contenuti della sorgente S vengano addizionati alla somma dei contenuti dell'accumulatore e al bit di riporto."

Le istruzioni ADC r e ADC M sono simili alle istruzioni ADD r e ADD M. L'unica differenza è che il bit di riporto è addizionato al bit meno significativo nel byte dell'accumulatore. Tutti i flag sono coinvolti da queste istruzioni. La locazione di memoria M è indirizzata dai contenuti della coppia di registri H,L.

ACI data

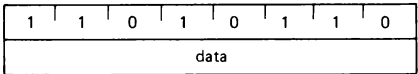
ACI data (Addizionare immediatamente con riporto)
(A) ← (A) + (byte 2) + (CY)
Il contenuto del secondo byte dell'istruzione e il contenuto del flag CY sono addizionati ai contenuti dell'accumulatore. Il risultato si trova nell'accumulatore.

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: Z,S,P,CY,AC

L'istruzione ACI data fa sì che la quantità di dati a 8 bit presente nel secondo byte dell'istruzione venga addizionata alla somma dei contenuti dell'accumulatore e al bit di riporto. L'istruzione coinvolge tutti e cinque i flag.

SUB r e SUB M

L'istruzione SUB r fa sì che i contenuti del registro sorgente S vengano sottratti dall'accumulatore. Il registro sorgente può essere uno qualunque dei registri universali B,C,D,E,H e L, l'accumulatore A, o M (i contenuti della memoria come vengono indirizzati dalla coppia di registri H,L). Tutti e cinque i flag sono coinvolti dall'esecuzione di questa istruzione. Se volete azzerare l'accumulatore, lo farà l'istruzione.

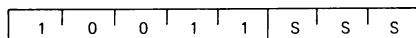
SUB A

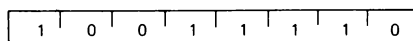
che ha un codice istruzione di 227.

SUI data

L'istruzione SUI data fa sì che la quantità di dati a 8 bit specificata nel secondo byte d'istruzioni venga sottratta dall'accumulatore.

Tutti e cinque i flag sono coinvolti.

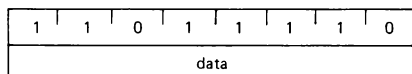
SBB r e SBB M

SBB r (Sottrare il registro con riporto negativo)
(A) ← (A) − (r) − (CY)
Il contenuto del registro r e il contenuto del flag CY vengono sottratti entrambi dall'accumulatore.
Il risultato si trova nell'accumulatore.

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: Z,S,P,CY,AC

SBB M (Sottrarre la memoria con riporto negativo)
(A) ← (A) − ((H) (L)) − (CY)
Il contenuto della locazione di memoria il cui indirizzo è contenuto nei registri H ed L e il contenuto del flag CY sono entrambi sottratti dall'accumulatore.
Il risultato si trova nell'accumulatore.

Cicli: 2
Stati: 7
Indirizzamento: reg. indiretto
Flag: Z,S,P,CY,AC

L'istruzione SBB r fa sì che i contenuti della sorgente S vengano sottratti dalla differenza dei contenuti dell'accumulatore e del bit di riporto negativo. Il registro sorgente può essere uno qualunque dei registri universali B,C,D,E,H e L; l'accumulatore A, o M, i contenuti di memoria indirizzati dalla coppia di registri H,L. Tutti e cinque i flag sono coinvolti dalle istruzioni SBB r e SBB/M.

SBI data

SBI data (Sottrarre immediatamente con riporto negativo)
(A) ← (A) − (byte 2) − (CY)
I contenuti del secondo byte dell'istruzione e i contenuti del flag CY vengono entrambi sottratti dall'accumulatore. Il risultato si trova nell'accumulatore.

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: Z,S,P,CY,AC

L'istruzione SBI data fa sì che la quantità di dati a 8 bit specificata nel secondo byte d'istruzioni venga sottratta dalla differenza dei contenuti dell'accumulatore e del bit di riporto negativo. Sono coinvolti tutti e cinque i flag.

È il caso di mostrare alcuni esempi delle varie operazioni di addizione e sottrazione. Considerate il programma seguente:

```
ADD B
ADD C
```

Se i contenuti iniziali del registro sono A = 00111110, B = 11100000 e C = 00101111, e se il bit di riporto fosse inizialmente zero, allora la sezione di codice suddetta produrrebbe il seguente risultato nell'accumulatore:

Bit di riporto		
0	0 0 1 1 1 1 1 0	Contenuti dell'accumulatore
	+ 1 1 1 0 0 0 0 0	Contenuti del registro B
1	0 0 0 1 1 1 1 0	Somma memorizzata nell'accumulatore
	+ 0 0 1 0 1 1 1 1	Contenuti del registro C
0	0 1 0 0 1 1 0 1	Somma memorizzata nell'accumulatore

Notate attentamente il comportamento del bit di riporto in questa situazione. Se non deriva nessun riporto dal bit più significativo (MSB) nell'accumulatore, il bit di riporto viene azzerato; se dal bit più significativo deriva un riporto nell'accumulatore durante l'addizione, il bit di riporto viene settato. Quando avete addizionato B all'accumulatore, avete avuto un riporto. Quando avete addizionato i contenuti di C alla somma, non c'è stato riporto. Il riporto di operazioni precedenti non viene utilizzato o "portato avanti". Adesso confrontiamo i risultati suddetti con il comportamento della seguente sezione di codice:

```
ADC B
ADC C
```

Supponiamo di avere gli stessi valori iniziali per i registri A, B, C e il bit di riporto. Otterreste i seguenti risultati:

Bit di riporto		
0	0 0 1 1 1 1 1 0	Contenuti dell'accumulatore
	+ 1 1 1 0 0 0 0 0	Contenuti del registro B
1	0 0 0 1 1 1 1 0	Somma memorizzata nell'accumulatore

Fin qui, non c'è differenza. Comunque, quando addizioniamo i contenuti del registro C alla somma suddetta, osserviamo una differenza:

	0 0 0 1 1 1 1 0	Somma memorizzata nell'accumulatore
	+ 1	Bit di riporto
	+ 0 0 1 0 1 1 1 1	Contenuti del registro C
0	0 1 0 0 1 1 1 0	Somma memorizzata nell'accumulatore

Ora consideriamo la seguente sezione di programma

```
SUB B
SUB C
```

per gli stessi valori iniziali dei registri A, B, C, e il bit di riporto. Notate che se ricavate un riporto negativo dall'MSB dell'accumulatore, il bit di riporto viene settato; se non vi è nessun riporto negativo, il bit di riporto viene azzerato. Dovreste perciò osservare quanto segue:

Bit di riporto

0	0 0 1 1 1 1 1 0	Contenuti dell'accumulatore
	− 1 1 1 0 0 0 0 0	Contenuti del registro B
1	0 1 0 1 1 1 1 0	Differenza memorizzata nell'accum.
	− 0 0 1 0 1 1 1 1	Contenuti del registro C
0	0 0 1 0 1 1 1 1	Differenza memorizzata nell'accum.

Ora eseguiamo operazioni di sottrazione usando le istruzioni SSB r,
SBB B
SBB C
Abbiamo i seguenti risultati:

Bit di riporto

0	0 0 1 1 1 1 1 0	Contenuti dell'accumulatore
	− 1 1 1 0 0 0 0 0	Contenuti del registro B
1	0 1 0 1 1 1 1 0	Differenza memorizzata nell'accum.

Quando eseguiamo l'operazione SBB C sottraiamo i contenuti del registro C dalla differenza fra il bit di riporto negativo e i contenuti dell'accumulatore:

	0 1 0 1 1 1 1 0	Differenza memorizzata nell'accum.
	− 1	
	− 0 0 1 0 1 1 1 1	Contenuti del registro C
0	0 0 1 0 1 1 1 0	Differenza memorizzata nell'accum.

Le istruzioni ADC r e SBB r vengono usate quando si eseguono operazioni aritmetiche in precisione doppie o triple. Un'operazione aritmetica in *doppia precisione* è un'operazione eseguita su due quantità di 16 bit per ottenere un risultato a 16 bit. Un'operazione in *tripla precisione* è quella eseguita su due quantità a 24 bit per ottenere un risultato a 24 bit. Gli esempi suddetti di operazioni di addizione e sottrazione sono stati gentilmente forniti dalla NEC Microcomputers, Inc. e sono tratti dal loro *"μCOM-8 Software Manual".*

DAA

DAA (Aggiustamento decimale dell'accumulatore)
Il numero a otto bit nell'accumulatore viene adattato per formare cifre decimali in codice binario a quattro bit per mezzo del seguente procediemnto:

1. Se il valore dei 4 bit meno significativi dell'accumulatore è maggiore di 9 o se il flag AC è settato, si addiziona 6 all'accumulatore.
2. Se il valore dei 4 bit più significativi dell'accumulatore è ora maggiore di 9, o se il flag CY è settato, si addizione 6 ai 4 bit più significativi dell'accumulatore.

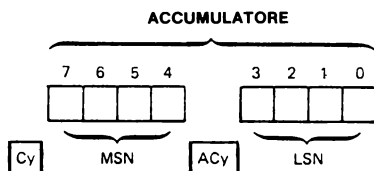
NOTA: Tutti i flag sono coinvolti.

Cicli: 1
Stati: 4
Flag: Z,S,P,CY,AC

Citiamo il *"µCOM-8 Software Manual":* Allo scopo di eseguire operazioni decimali in codice binario (BCD), occorre un'istruzione speciale. Quando la CPU dell'8080A esegue un'operazione aritmetica, produce il risultato in codice binario. Lavorando in BCD, questo non produce il risultato corretto. Per rimediarvi, si usa un'istruzione DAA. DAA sta per Aggiustamento Decimale, che è esattamente ciò che fa DAA. L'istruzione DAA tratta l'accumulatore a 8 bit come due accumulatori a 4 bit. Tramite l'uso di un flag non testabile conosciuto come l'Auxiliary Carry (Riporto Ausiliario), l'operazione DAA adatta il risultato di un'operazione di addizione binaria al BCD impaccato .

"Per l'addizione, l'istruzione DAA dà luogo all'operazione seguente. Se il Carry Ausiliario è posizionato su uno o se il digit meno significativo (LSD) è maggiore di 9, si addiziona sei al digit meno significativo. Poi, se il flag di riporto è posizionato su uno o il digit più significativo è maggiore di 9, si addiziona sei al digit più significativo (MSD). Il termine "digit" viene cosí definito:

Digit — Un gruppo di quattro bit contigui che solitamente rappresentano una cifra decimale in codice binario (BCD).

Il digit meno significativo (LSD), il digit più significativo (MSD), l'accumulatore, il flag di riporto ausiliario (ACy)n e il flag di riporto (Cy) possono essere rappresentati nel modo seguente:

Supponiamo, come mostra un esempio del *"µCOM-8 Software Manual"*, che l'accumulatore contenga la rappresentazione BCD di 75 (MSD = 0111 e LSD = 0101) e che il registro B contenga la rappresentazione BCD di 38 (MSD = 0011 e LSD = 1000) e il flag di riporto sia a livello logico 0. L'istruzione, ABC B, produce il seguente risultato nell'accumulatore:

Bit di riporto	Bit di riporto ausiliario		
0	–	0 1 1 1 0 1 0 1	Contenuti dell'accumulatore
		+ 0 0 1 1 1 0 0 0	Contenuti del registro B
0	0	1 0 1 0 1 1 0 1	Somma memorizzata nell'accumulatore

Con il Riporto Ausiliario, se l'istruzione produce un riporto dal bit 3 nel bit 4 del valore risultante, il flag di Riporto Ausiliario è settato; altrimenti viene resettato. Nell'esempio precedente, non c'è riporto dal bit 3 e nel bit 4, e il bit di Riporto Ausiliario è zero dopo l'operazione.

Il comando DAA trova ACy resettato a 0 e LSDN = 1101. Dato che l'LSD è maggiore di 9, vi si addiziona sei e il risultato è 0011. Dato che l'MSD è maggiore di nove, vi si addiziona sei e il risultato è 0001. Il risultato finale dopo l'operazione. DAA è

1	1	0 0 0 0 0 0 1 1	Somma aggiustata decimale
1	–	1 3	Numero decimale

che è equivalente al numero decimale. 103. L'operazione DAA può essere scritta nel modo seguente:

Bit di riporto	Bit di riporto ausiliario			
0	0	1 0 1 0	1 1 0 1	Somma
		+ 0 1 1 0	+ 0 1 1 0	Operazione DAA
1	1	0 0 0 0	0 0 1 1	Risultato dell'operazione DAA
1	1	0 0 0 1	0 0 1 1	BCD
1	-	0	3	Numero decimale

Così, 75 + 38 = 103.

"Nell'operazione attuale, l'adattamento DAA è fatto in parallelo, piuttosto che in serie, come illustrato. Comunque, questa spiegazione seriale (ricavata dal *"μCOM-8 Software Manual"* della NEC Microcomputers, Inc.) è più semplice da capire ed illustra meglio l'adattamento. *L'istruzione DAA dovrebbe seguire immediatamente un'operazione di addizione o di incremento, dato che alcune istruzioni dell'8080A alterano lo stato del flag di riporto ausiliario. Un'alterazione di questo tipo potrebbe rivelarsi in risultati non corretti.".*

C'è una differenza importante fra il chip Intel 8080A e il chip equivalente, il chip μCOM-8 della NEC Microcomputers, Inc. Il chip μCOM-8 ha un flag non provabile in più, chiamato Subtract. Citiamo dal Manuale NEC: "Per l'addizione, il flag Sub viene posizionato su zero . . . . Per la sottrazione, Sub è posizionato su uno, dando luogo all'operazione DAA successiva. Se ACy è posizionato su uno (vi è stato un riporto negativo) si sottrae sei da LSD. Poi, se Cy è posizionato su uno (vi è stato un riporto negativo) si sottrae sei da MSD. L'uso di un'istruzione DAA immediatamente dopo un'operazione su due byte in formato BCD impaccato aggiusta il risultato a due cifre BCD e ad un riporto (sia negativo che positivo) in formato BCD
Notate che le operazioni DAA vengono eseguite direttamente dopo la sottrazione, *eliminando la necessità dell'aritmetica di complemento per 100 per la sottrazione".*

Se state eseguendo un notevole numero di manipolazioni BCD, dovrebbe interessarvi il chip COM-8 piuttosto che il chip 8080A. Comunque, ciò sarebbe valido nel caso aveste bisogno di far lavorare il microcomputer alla massima velocità. Con istruzioni aggiuntive, l'8080A può facilmente assolvere lo stesso compito di produrre un formato BCD dopo una sottrazione.

INR r e INR M

INR r (Incrementare il registro)
(r) ← (r) + 1
Il contenuto del registro r è incrementato di uno.
Nota: tutti i flag, eccetto CY, sono coinvolti.

0	0	D	D	D	1	0	0

Cicli: 1
Stati: 5
Indirizzamento: registro
Flag: Z,S,P,AC

INR M (Incrementare la memoria)
((H) (L)) ← ((H) (L)) + 1
Il contenuto della posizione di memoria il cui indirizzo è contenuto nei registri H ed L è incrementato di uno.
Nota: tutti i flag, eccetto CY, sono coinvolti.

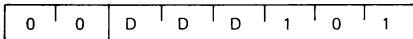
0	0	1	1	0	1	0	0

Cicli: 3
Stati: 10
Indirizzamento: reg. indiretto
Flag: Z,S,P,AC

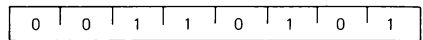
L'istruzione INR r fà sì che un uno venga addizionato al registro di destinazione D. Il registro di destinazione può essere uno qualunque dei registri universali B, C, D, E, H, e L; l'accumulatore A; o M, i contenuti di memoria così come vengono indirizzati dalla coppia di registri H,L. Tutti i flag sono coinvolti, ad eccetto di quello di riporto (CY).

DCR r e DCR M

DCR r (Decrementare il registro)
(r) ← (r) − 1
Il contenuto del registro r è decrementato di uno.
Nota: Tutti i flag, **eccetto CY,** sono coinvolti.

0	0	D	D	D	1	0	1

Cicli: 3
Stati: 5
Indirizzamento: reg. indiretto
Flag: Z,S,P,AC

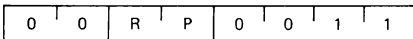
DCR M (Decrementare la memoria)
((H) (L)) ← ((H) (L)) − 1
Il contenuto della locazione di memoria il cui indirizzo è contenuto nella coppia H ed L è decrementato di uno.
Nota: Tutti i flag, **eccetto CY,** sono coinvolti.

0	0	1	1	0	1	0	1

Cicli: 3
Stati: 10
Indirizzamento: reg. indiretto
Flag: Z,S,P,AC

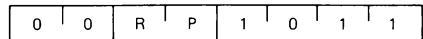
L'istruzione DCR r fa sì che venga sottratto un uno dal registro di destinazione D. Il registro di destinazione può essere uno qualunque dei registri universali B, C, D, E, H e L; l'accumulatore A; o M, i contenuti di memoria così come vengono indirizzati dalla coppia di registri H, L. Solo quattro dei cinque flag sono coinvolti; il flag di riporto resta invariato.

INX rp e DCX rp

INX rp (Incrementare la coppia di registri)
(rh) (rl) ← (rh) (rl) + 1
Il contenuto della coppia di registri rp è incrementato di uno.
Nota: **Nessun flag è coinvolto.**

0	0	R	P	0	0	1	1

Cicli: 1
Stati: 5
Indirizzamento: registro
Flag: nessuno

DCX rp (Decrementare la coppia di registri)
(rh) (rl) ← (rh) (rl) − 1
Il contenuto della coppia di registri rp è decrementato di uno.
Nota: **Nessun flag è coinvolto.**

0	0	R	P	1	0	1	1

Cicli: 1
Stati: 5
Indirizzamento: registro
Flag: nessuno

INX rp fa sì che la coppia di registri specificata da RP venga incrementata di uno; DCX rp fa sì che la coppia di registri specificata da RP venga decrementata di uno. RP può essere la coppia di registri specificata da B, D, o H (che corrispondono a BC, DE, o HL) o lo stack pointer a 16 bit specificato da SP. INX e DCX non coinvolgono nessun bit di flag; essi di solito non vengono usati nelle operazioni aritmetiche, dato che il loro uso principale sta nell'incrementare e decrementare gli indirizzi di memoria a 16 bit.

DAD rp

DAD rp (Addizionare la coppia di registri ad H ed L)
(H) (L) ← (H) (L) + (rh) (rl)
Il contenuto della coppia di registri rp è addizionato al contenuto della coppia di registri H ed L.
Il risultato si trova nella coppia di registri H ed L.
Nota: **solo il flag CY è coinvolto.** Esso è settato se vi è un riporto dall'addizione in doppia precisione; altrimenti è resettato.

0	0	R	P	1	0	0	1

Cicli: 6
Stati: 10
Indirizzamento: registro
Flag: CY

Secondo il Manuale NEC: "Mentre le istruzioni INX e DCR permettono l'incremento e il decremento della coppia di registri, l'istruzione DAD, doppio ADD, permette di addizionare insieme le coppie di registri. DAD rp fa sì che la coppia di registri specificata da RP venga addizionata ai contenuti della coppia di registri HL, e il risultato rimane nella coppia HL. Il flag di riporto è il solo flag di stato coinvolto dall'istruzione DAD. . Le istruzioni INX, DCX e DAD permettono il calcolo della ricerca tabellare". Sono usate anche per l'indirizzamento modificato da indice e la manipolazione dei dati in archivio.

CMP r e CMP M

CMP r (Confrontare il registro)
(A) - (r)
Il contenuto del registro r è sottratto dall'accumulatore. L'accumulatore resta invariato. I flag vengono posizionati come risultato della sottrazione.
Il flag Z è posizionato su 1 se (A) = (r).
Il flag CY è posizionato su 1 se (A) < (r).

1	0	1	1	1	S	S	S

Cicli: 1
Stati: 4
Indirizzamento: registro
Flag: Z,S,P,CY,AC

CMP M (Confrontare la memoria)
(A) − (H) (L)
Il contenuto della posizione di memoria il cui indirizzo è contenuto nei registri H ed L viene sottratto dall'accumulatore. L'accumulatore resta invariato.
I flag sono posizionati come risultato della sottrazione.
Il flag Z è posizionato su 1 se (A) = ((H) (L))
Il flag CY è posizionato su 1 se (A) < ((H) (L)).

1	0	1	1	1	1	1	0

Cicli: 2
Stati: 7
Indirizzamento: reg. indiretto
Flag: Z,S,P,CY,AC

Citiamo il *"µCOM-8 Software Manual"*: "CMP r e CMP M sono usati per confrontare due *dati senza alterarli*. CMP r confronta i contenuti dell'accumulatore con uno dei singoli registri B,C,D,E,H e L; l'accumulatore A; o M, la posizione di memoria indirizzata dalla coppia di registri H,L.

L'istruzione non coinvolge nessuno dei registri di dati, ma coinvolge i quattro bit di flag Carry, Zero, Sign e Parity. Le istruzioni di confronto eseguono una sottrazione interna della sorgente S dall'accumulatore. I flag vengono settati sulla base di quello che sarebbe stato il risultato della sottrazione. Così Zero è settato se le quantità erano uguali, Sign è settato se il risultato era negativo (il bit più significativo è a livello logico 1), Parity è settato se il risultato è di parità, e Carry è settato se c'è un riporto negativo dal bit 7 (dati sorgente maggiori dei dati dell'accumulatore)."

"Così, in ogni caso:

Carry è settato se vi è un riporto negativo; altrimenti è resettato
Sign è settato uguale al MSB del risultato;
Zero è settato se il risultato è zero; altrimenti è resettato.
Parity è settato se la parità del risultato è even; altrimenti è resettato.

Le istruzioni di confronto sono usate nel modo migliore per i confronti aritmetici senza segno (numeri compresi nella fascia di valori da 0 a 255_{10}), chiamati anche confronti logici o di caratteri. In questo caso, i risultati per i flag Zero e Carry possono essere interpretati nel modo seguente:

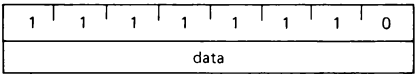
Risultato dell'operazione di confronto

Flag di Zero	Flag di Riporto	Rapporto fra accumulatore e registro
1	X	Accumulatore = registro
X	1	Accumulatore < registro
1	1	Accumulatore ⩽ registro
0	0	Accumulatore > registro
X	0	Accumulatore ⩾ registro

NOTA: X = non ha importanza

Così, le relazioni { = , <, ⩾ } possono essere provate usando una sola istruzione di salto, mentre { ⩽, > } ne richiedono due. Notate che se gli operandi vendono invertiti, > rimpiazza ⩽ e < rimpiazza ⩾."

CPI data

CPI data (Confrontare in modo immediato)
(A) — (byte 2)
Il contenuto del secondo byte dell'istruzione è sottratto dall'accumulatore. I flag sono posizionati dal risultato della sottrazione. Il flag Z è settato ad 1 se (A) = (byte 2). Il flag CY è settato ad 1 se (A) < (byte 2).

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: Z,S,P, CY, AC

L'istruzione CPI data è un'operazione immediata che confronta i contenuti dell'accumulatore con la quantità di 8 bit nel secondo byte dell'istruzione. L'istruzione coinvolge tutti e cinque i flag, ma solo quattro di questi producono risultati utili. I flag vengono settati o azzerati sulla base di quello che avrebbe dovuto essere il risultato della sottrazione. *I contenuti dell'accumulatore restano invariati.* Per ulteriori particolari, vedere la discussione precedente sull'istruzione CMP r.

Si può dedurre che le istruzioni CMP r e CPI data sono operazioni logiche e non aritmetiche. Dal momento che viene eseguita un'operazione aritmetica - la sottrazione - le includeremo nel gruppo delle operazioni aritmetiche. Lo scopo delle istruzioni di confronto è produrre decisioni che vengono riflesse negli stati logici dei bit di flag.

GRUPPO LOGICO

Questo gruppo di istruzioni esegue operazioni logiche, cioè Booleane, su dati nei registri e nella memoria e sui flag di condizione. Salvo diversa indicazione, tutte le istruzioni di questo gruppo coinvolgono i flag Zero, Sign, Parity, Carry, Auxiliary Carry e Carry secondo le regole standard.

ANA r e ANA M

ANA r (AND del registro)
(A) ← (A) ∧ (r)
Il contenuto del registro r è addizionato in maniera logica al contenuto dell'accumulatore. Il risultato si trova nell'accumulatore.
I flag CY e AC vengono azzerati.

1	0	1	0	0	S	S	S

Cicli: 1
Stati: 4
Indirizzamento registro
Flag: Z,S,P,CY,AC

ANA M (AND della memoria)
(A) ← (A) ∧ ((H) (L))
I contenuti della locazione di memoria il cui indirizzo è contenuto nei registri H ed L è addizionato in maniera logica al contenuto dell'accumulatore.
Il risultato si trova nell'accumulatore.
I flag CY e AC vengono azzerati.

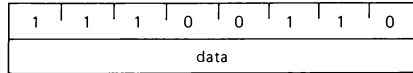
Cicli: 2
Stati: 7
Indirizzamento: reg. indiretto
Flag: Z,S,P,CY,AC

L'istruzione ANA r esegue un *AND logico bit a bit parallelo* dei contenuti dell'accumulatore e dei contenuti del registro sorgente S. Il registro sorgente può essere uno qualunque dei registri generali B, C, D, E, H e L; l'accumulatore A; o M, i contenuti della posizione di memoria indirizzati dalla coppia di registri H,L. Per esempio, l'operazione ANA B esegue un'operazione AND logica bit a bit con i contenuti del registro B e dell'accumulatore. Il caso particolare di

ANA A

azzera il flag di Carry e fa sì che il flag di Zero venga settato se il risultato è zero, azzerato se il risultato non è zero. Tutti i flag sono coinvolti dall'istruzione ANA r. Dal momento che A·A = A, i dati nell'accumulatore non sono cambiati. Questo è un "trucco" per azzerare il flag di riporto o semplicemente per testare se l'accumulatore ha valore zero.

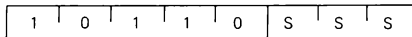
ANI data

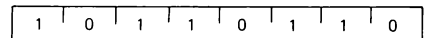
ANI data (AND in modo immediato)
(A) ← (A) ∧ (byte 2)
Il contenuto del secondo byte dell'istruzione è addizionato in maniera logica ai contenuti dell'accumulatore. Il risultato si trova nell'accumulatore. **I flag CY e AC vengono azzerati.**

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: Z,S,P,CY,AC

L'istruzione ANI data esegue un AND logico bit a bit dei contenuti dell'accumulatore con i contenuti del secondo byte dell'istruzione. Tutti i flag vengono coinvolti da questa istruzione.

ORA r e ORA M

ORA r (OR del registro)
(A) ← (A) V (r)
Viene eseguita un'operazione di OR inclusivo con il contenuto del registro r e il contenuto dell'accumulatore. Il risultato si trova nell'accumulatore.
I flag CY e AC vengono azzerati.

1	0	1	1	0	S	S	S

Cicli: 1
Stati: 4
Indirizzamento: registro
Flag: Z,S,P,CY,AC

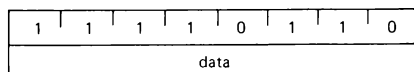
ORA M (OR della memoria)
(A) ← (A) V ((H) (L))
Viene eseguita un'operazione di OR inclusivo con il contenuto della locazione di memoria, il cui indirizzo è contenuto nei registri H ed L, e il contenuto dell'accumulatore.
Il risultato si trova nell'accumulatore.
I flag CY e AC vengono azzerati.

Cicli: 2
Stati: 7
Indirizzamento: reg. indiretto
Flag: Z,S,P,CY,AC

L'istruzione ORA r esegue un *OR logico bit a bit parallelo* dei contenuti dell'accumulatore e dei contenuti del registro sorgente S. Il registro sorgente può essere uno qualunque dei registri universali B, C, D, E, H e L; l'accumulatore A; o M, i contenuti della locazione di memoria indirizzata dalla coppia di registri H,L.
Il comando,

ORA A

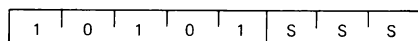
Che ha il codice di istruzione ottale 267, è un sistema conveniente per azzerare il flag di riporto senza coinvolgere nient'altro.
Sia ORA r che l'istruzione a due byte relativa, ORI data, azzerano il flag di riporto e fanno sì che il flag di Zero venga settato se il risultato è zero, azzerato se il risultato non è zero.

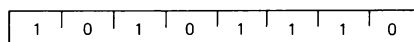
ORI data

ORI data ((OR in modo immediato)
(A) ← (A) ∀ (byte 2)
Viene eseguita un'operazione di OR inclusivo del contenuto del secondo byte dell'istruzione con il contenuto dell'accumulatore. Il risultato si trova nell'accumulatore. **I flag CY e AC vengono azzerati.**

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: Z,S,P,CY,AC

L'istruzione ORI data esegue un OR logico bit a bit dei contenuti dell'accumulatore con i contenuti del secondo byte dell'istruzione. Tutti i flag sono coinvolti da questa istruzione.

XRA r e XRA M

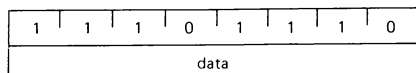
XRA r (OR esclusivo del registro)
(A) ← (A) ∀ (r)
Viene eseguita un'operazione di OR esclusivo del contenuto del registro r con il contenuto dell'accumulatore. Il risultato si trova nell'accumulatore. **I flag CY e AC vengono azzerati.**

Cicli: 1
Stati: 4
Indirizzamento: registro
Flag: Z,S,P,CY,AC

XRA M (OR esclusivo della memoria)
(A) ← (A) ∀ ((H) (L))
Viene eseguita un'operazione di OR esclusivo del contenuto della locazione di memoria il cui indirizzo è contenuto nei registri H ed L con il contenuto dell'accumulatore.
Il risultato si trova nell'accumulatore.
I flag CY e AC vengono azzerati.

Cicli: 2
Stati: 7
Indirizzamento: reg. indiretto
Flag: Z,S,P,CY,AC

L'istruzione XRA r esegue un *OR esclusivo logico bit a bit* dei contenuti dell'accumulatore e del registro sorgente S. Il registro sorgente può essere uno qualunque dei registri universali B, C, D, E, H ed L; l'accumulatore A; o M, la locazione di memoria indirizzata dalla coppia di registri H,L. Tutti i flag sono coinvolti da questa istruzione.

XRI data

L'istruzione XRI data esegue un OR esclusivo logico bit a bit dei contenuti dell'accumulatore con i contenuti del secondo byte dell'istruzione. Tutti i flag sono coinvolti dall'istruzione.

XRI data (OR esclusivo in modo immediato)
(A) ← (A) ⊻ (byte 2)
Viene eseguita un'operazione di OR esclusivo del contenuto del secondo byte dell'istruzione con il contenuto dell'accumulatore. Il risultato si trova nell'accumulatore. **I flag CY e AC vengono azzerati.**

Cicli: 2
Stati: 7
Indirizzamento: immediato
Flag: Z,S,P,CY,AC

Citiamo il *μCOM-8 Software Manual* della NEC Microcomputers, Inc.: "L'istruzione logica suddetta verrà usata per implementare una tecnica di programmazione conosciuta come *"masking"* (mascheratura). Si tratta di una tecnica per mezzo della quale i bit di un operando vengono selettivamente modificati per essere usati in una successiva operazione. Vi sono tre tipi generali di "masking":

- Azzerare tutti i bit sui quali non si opera
- Settare tutti i bit sui quali non si opera (usati raramente)
- Lasciare inalterati tutti i bit sui quali non si opera.

I primi due vengono chiamati *"masking esclusivo"*, e il terzo *"masking inclusivo"*. Ad esempio, supponiamo che l'accumulatore contenga il seguente valore,

bit: 7 6 5 4 3 2 1 0
1 1 0 1 1 0 1 1 0 Contenuti dell'accumulatore

Per provare se il bit 3 è zero o uno e simultaneamente azzerare gli altri bit, l'accumulatore è mascherato con 00001000. Usando l'istruzione,

ANI
010

l'accumulatore conterrà zeri con il flag di Zero settato se il bit di 3 era stato uno zero, e conterrà 010, in codice ottale, con il flag di Zero azzerato se il bit 3 era stato uno".

"Allo scopo di posizionare il bit 3 su uno e lasciare gli altri bit inalterati, la stessa configurazione di bit viene usata, assieme all'istruzione,

ORI
010

Il risultato in questo caso è 11011110 nell'accumulatore".

"Allo scopo di posizionare il bit 3 su zero e lasciare gli altri bit inalterati, l'accumulatore è addizionato con 1111011, il complemento della maschera del primo esempio. Con l'istruzione.

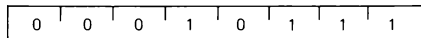
ANI
367

il risultato dell'accumulatore è 11010110. Queste sono le operazioni di manipolazione dei bit usate più comunemente, dato che la mascheratura viene eseguita in una sola istruzione.

Ne sono possibili molte altre, ma spesso richiedono più di un'istruzione per l'implementazione."

ISTRUZIONI DI ROTAZIONE

Tutte le istruzioni di rotazione 8080A sono riassunte nello schema seguente:

RAL e RAR

L'istruzione RAL, o "Ruotare l'Accumulatore a Sinistra", fa sì che l'accumulatore faccia ruotare tutti i bit di una posizione verso sinistra attraverso il bit di riporto, cioè una rotazione a 9 bit: Il bit 7 si trasferisce al flag di riporto, il bit di riporto si trasferisce al bit 0, il bit 0 si trasferisce al bit 1, il bit 1 si trasferisce al bit 2, e così via, come mostra la pagina precedente.

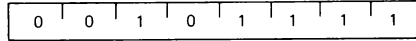
L'istruzione RAR, o "Ruotare l'Accumulatore a Destra", fa sì che l'accumulatore faccia ruotare tutti i bit di una posizione verso destra attraverso il bit di riporto, cioè una rotazione a 9 bit. Il bit 0 si trasferisce al flag di riporto, il bit di riporto si trasferisce al bit 7, il bit 7 si trasferisce al bit 6, e così via, come mostra la pagina precedente.

RLC e RRC

L'istruzione RLC, o "Ruotare a Sinistra Circolarmente", fa ruotare l'accumulatore di un bit verso sinistra e nel flag di riporto, come mostra lo schema di pagina precedente.

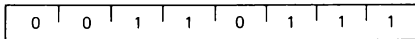
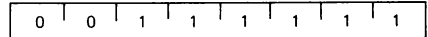
L'istruzione RRC, o "Ruotare a Destra Circolarmente", fa ruotare l'accumulatore di un bit verso destra e nel flag di riporto, come mostra lo schema di pagina precedente.

In entrambe queste istruzioni, l'informazione originale che appare sul flag di riporto, viene perduta.

CMA

L'istruzione CMA complementa i contenuti dell'accumulatore senza coinvolgere nessuno dei bit di flag. Per esempio, se l'accumulatore contenesse 11010001, l'istruzione

CMA

lo convertirebbe in 00101110. Viene complementato ogni singolo bit.

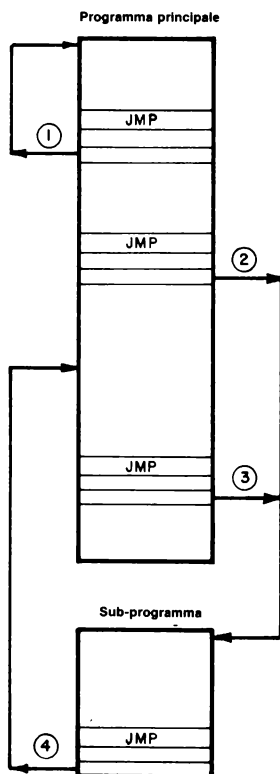
STC e CMC

L'istruzione STC setta il flag di riporto sul livello logico 1; l'istruzione CMC complementa il flag di riporto. Non è coinvolto nessun altro bit di flag.

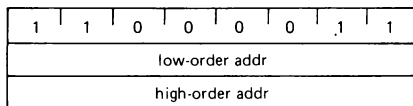
GRUPPO DI SALTO

Questo gruppo di istruzioni altera il normale flusso sequenziale di un programma. *I flag di condizione non sono coinvolti da nessuna istruzione di questo gruppo.* I due tipi di istruzioni di salto sono incondizionato e condizionato. I trasferimenti incondizionati eseguono semplicemente l'operazione specificata sul registro PC, il contatore di programma. I trasferimenti condizionati esaminano lo stato di uno dei quattro flag di processo - Zero, Sign, Parity, o Carry - per determinare se l'operazione di salto specifica deve essere eseguita. Le condizioni che possono essere specificate sono le seguenti:

CONDIZIONE	CCC
NZ - non zero (Z=0)	000
Z - zero (Z =1)	001
NC - nessun riporto (CY = 0)	010
C - riporto (CY = 1)	011
PO - parità dispari (P = 0)	100
PE - parità pari (P = 1)	101
P - più (S = 0)	110
M - meno (S = 1)	111

NOTA: CCC è il codice a tre bit per la condizione dei flag

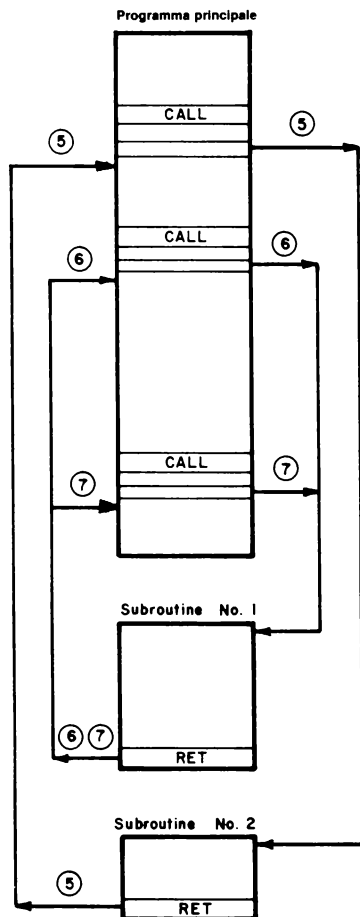
JMP addr

JMP addr (Salto)
(PC) ← (byte 3) (byte 2)
Il controllo è trasferito all'istruzione il cui indirizzo è specificato nei byte 2 e 3 di questa istruzione.

1	1	0	0	0	0	1	1
low-order addr							
high-order addr							

Cicli: 3
Stati: 10
Indirizzamento: immediato
Flag: nessuno

Il *contatore di programma (program counter)* è il registro a 16 bit nel microprocessore 8080A che contiene l'indirizzo di memoria del byte di istruzione successivo che deve essere eseguito in un programma. L'istruzione JMP addr è semplicemente un'istruzione di trasferimento di byte, nella quale il secondo e il terzo byte istruzioni vengono trasferiti direttamente

al contatore di programma. Non sono incluse operazioni aritmetiche nè logiche, e non è coinvolto nessun bit di flag. L'istruzione JMP è un'istruzione a tre byte che contiene l'indirizzo di memoria a 16 bit al quale viene trasferito il controllo di programma. Potete saltare avanti o indietro a tutte le 65.536 possibili locazioni di memoria. Il microprocessore non ricorda il punto dal quale è saltato, in netto contrasto con il comportamento delle istruzioni CALL e RET descritte più avanti.

Il comportamento dell'istruzione JMP può essere compreso con l'aiuto dello schema mostrato precedentemente. La prima istruzione JMP, ①, è un salto indietro che crea un loop. JMP② e JMP③ trasferiscono il controllo di programma al sottoprogramma. L'uscita del sottoprogramma è designata dall'istruzione JMP④.

CALL addr e RET

CALL addr (Richiamo)
((SP) − 1) ← (PCH)
((SP) − 2) ← (PCL)
(SP) ← (SP) − 2
(PC) ← (byte 3) (byte 2)
Gli otto bit di ordine più elevato dell'indirizzo dell'istruzione successiva sono posti nella locazione di memoria il cui indirizzo è uno meno del contenuto del registro SP.
Gli otto bit di ordine inferiore dell'indirizzo dell'istruzione successiva sono posti nella locazione di memoria il cui indirizzo è due meno del contenuto del registro SP. Il contenuto del registro SP è decrementato di 2. Il controllo è trasferito all'istruzione il cui indirizzo è specificato nei byte 3 e 2 dell'istruzione in corso.

1	1	0	0	1	1	0	1
low-order addr							
high-order addr							

Cicli: 5
Stati: 17
Indirizzamento: immediato/reg. indiretto
Flag: nessuno

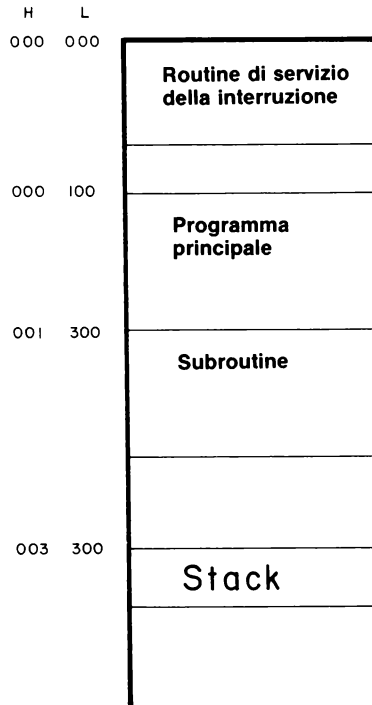
RET (Ritorno)
(PCL) ← ((SP));
(PCH) ← ((SP) + 1);
(SP) ← (SP) + 2;

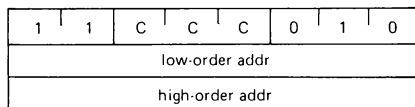
Il contenuto della locazione di memoria il cui indirizzo è specificato nel registro SP, è posto negli otto bit di ordine inferiore del registro PC. Il contenuto della locazione di memoria il cui indirizzo è uno più del contenuto del registro SP, è spostato agli otto bit di ordine più elevato del registro PC.
Il contenuto del registro SP è incrementato di 2.

1	1	0	0	1	0	0	1

Cicli: 3
Stati: 10
Indirizzamento: reg. indiretto
Flag: nessuno

Può darsi che spesso vogliate "saltar fuori" da un programma principale, ma ritornarvi più tardi. Per farlo, dovete non solo conoscere la vostra nuova destinazione, ma dovete anche ricordare in qualche modo la vostra locazione originale. A questo scopo, avete due tipi di istruzioni, richiamo di subroutine e rientro dalla subroutine. Qui parleremo delle istruzioni incondizionate CALL addr e RET. Citiamo il Manuale della NEC Microcomputers: "L'istruzione CALL trasferisce il controllo ad una subroutine. L'istruzione CALL addr salva il contatore di programma incrementato nello *stack* e pone l'indirizzo nel contatore di programma. Lo stack è un blocco di memoria lettura/scrittura indirizzato da uno speciale registro a 16 bit conosciuto come "stack pointer" che può essere caricato dall'utente (LXI SP, data 16). Lo stack opera come una memoria "last-in-first-out" (LIFO), con il registro stack pointer che indirizza il dato più recente posto nello stack. L'istruzione di rientro fa sì che il puntatore dello stack venga posto nel contatore di programma. Così un'istruzione di richiamo (CALL) trasferisce il controllo di programma dal programma principale nella subroutine e un'istruzione RET ritrasferisce il controllo al programma principale."

La posizione dello *stack* è di solito agli indirizzi più alti nella memoria disponibile di un microcomputer basato sull'8080A. Nel diagramma seguente, lo stack è ad una certa distanza dal programma principale e dalle subroutines.

JNZ, JZ, JNC, JC, JPO, JPE, JP, e JM addr

Jcondition addr (Salto condizionato)
Se (CCC),
(PC) ← (byte 3) (byte 2)
Se la condizione specificata è vera, il controllo è trasferito all'istruzione il cui indirizzo è specificato nei byte 3 e 2 dell'istruzione in corso; altrimenti, il controllo continua sequenzialmente.

Cicli: 3
Stati: 10
Indirizzamento: immediato
Flag: nessuno

In un'istruzione di salto condizionato, se la condizione è soddisfatta, il secondo e il terzo byte dell'istruzione vengono trasferiti al contatore di programma e si verifica un salto. Se la condizione non è soddisfatta, non avviene nessun cambiamento nel contatore di programma; il controllo di programma passa all'istruzione che segue immediatamente il salto.

Le varie condizioni possono essere riassunte come segue:

NZ: Il risultato a 8 bit nell'operazione aritmetica o logica immediatamente precedente Non è uguale a Zero, cioè il flag Zero è stato azzerato.

Z: Il risultato a 8 bit dell'operazione aritmetica o logica immediatamente precedente è uguale a Zero, cioè il flag di Zero è stato settato.

NC: Il risultato a 8 bit dell'operazione aritmetica o logica immediatamente precedente non produce nessun riporto (No Carry) dal bit più significativo; o, il flag di riporto è stato azzerato.

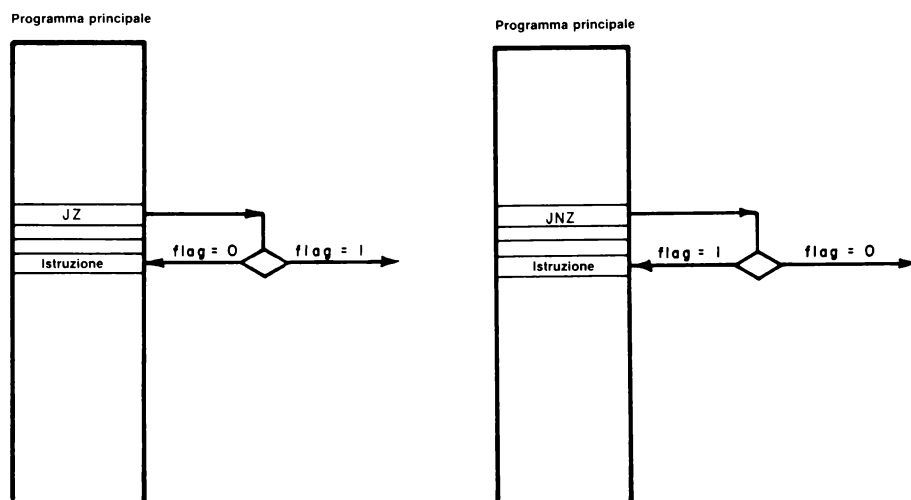
C: Il risultato a 8 bit dell'operazione aritmetica o logica immediatamente precedente produce un riporto (Carry) dal bit più significativo; o, il flag di riporto è settato.

PO: Il risultato a 8 bit dell'operazione aritmetica o logica immediatamente precedente ha una Parità che è "Odd", cioè il flag di parità è stato azzerato.

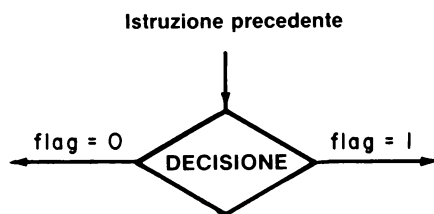
PE: Il risultato a 8 bit dell'operazione aritmetica o logica immediatamente precedente ha una Parità che è "Even", cioè il flag di parità è stato settato.

P: Il risultato a 8 bit dell'operazione aritmetica o logica immediatamente precedente produce un MSB che ha un segno Più, cioè il flag di segno è stato azzerato.

M: Il risultato a 8 bit dell'operazione aritmetica o logica immediatamente precedente produce un MSB che ha un segno Meno, cioè il flag di segno è stato settato.

Il valore di CCC che corrisponde ad ognuna delle condizioni è stato mostrato parecchie pagine indietro. Il comportamento di due delle istruzioni condizionate, JNZ e JZ, può essere compreso con l'aiuto del diagramma seguente:

Nell'istruzione JNZ, il salto avviene solo se il risultato a 8 bit di un'operazione aritmetica o logica Non è Zero. Il simbolo di scelta logica,

Istruzione precedente

flag = 0 — DECISIONE — flag = 1

che viene usato nei flow-chart, indica che ciò che succederà poi dipende dallo stato del flag di Zero. Per JNZ, avviene un salto se il flag di Zero è azzerato, cioè a livello logico 0. Per JZ, avviene un salto se il risultato a 8 bit è uguale a zero; in tal caso, il flag di Zero è a livello logico 1.

È possibile confondersi a proposito delle condizioni NZ e Z. Notate che NZ e Z si riferiscono al risultato a 8 bit di un'operazione, non allo stato logico del flag di Zero. NZ significa che il risultato a 8 bit di un'operazione non è zero; Z significa che il risultato a 8 bit di un'operazione è zero (sebbene il flag di Zero sia a livello logico 1). Nel discutere l'argomento, abbiamo tentato di dimostrare che una condizione può essere vista come il risultato a 8 bit di un'operazione aritmetico/logica (NZ,Z,NC,C,PO,PE,P,M) o come lo stato logico dei singoli flag che prova il risultato di un'operazione aritmetico/logica.

Preferiamo usare il risultato a 8 bit di un'operazione ALU, compresi i simboli letterali NZ,Z,NC, ecc. Speriamo di non avervi messo in confusione.

CNZ, CZ, CNC, CC, CPO, CPE, CP e CM addr

Condition addr (Richiamo condizionato)
Se (CCC),
((SP) − 1) ← (PCH)
((SP) − 2) ← (PCL)
(SP) ← ((SP) − 2
(PC) ← (byte 3) (byte 2)
Se la condizione specificata è vera, vengono eseguite le azioni specificate nell'istruzione di CALL (vedi sopra); altrimenti, il controllo continua sequenzialmente.

Cicli: 3/5
Stati: 11/17
Indirizzamento: immediato/reg.indiretto
Flag: nessuno

In un'istruzione di richiamo *(call)* condizionato, se la condizione è soddisfatta, viene richiamata la subroutine della locazione di memoria data nel secondo e nel terzo byte di istruzioni. I contenuti del contatore di programma sono posti sullo stack, in modo che un'istruzione di rientro potrà rimandare il controllo di programma all'istruzione che segue immediatamente l'istruzione di salto condizionato.

Se la condizione non è soddisfatta, l'esecuzione del programma passa all'istruzione che segue immediatamente l'istruzione di richiamo condizionato.

RNZ, RZ, RNC, RC, RPO, RPE, RP, e RM

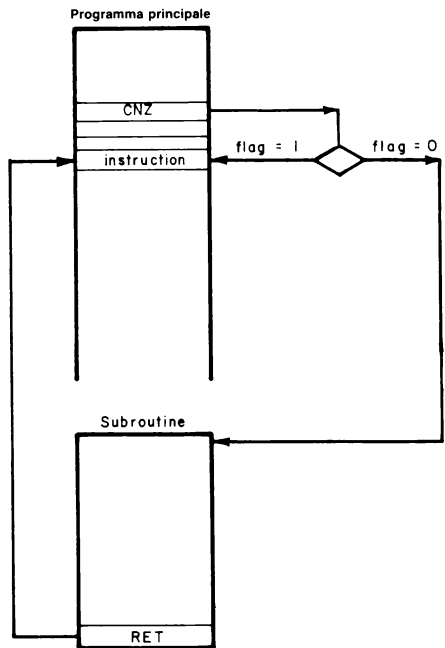
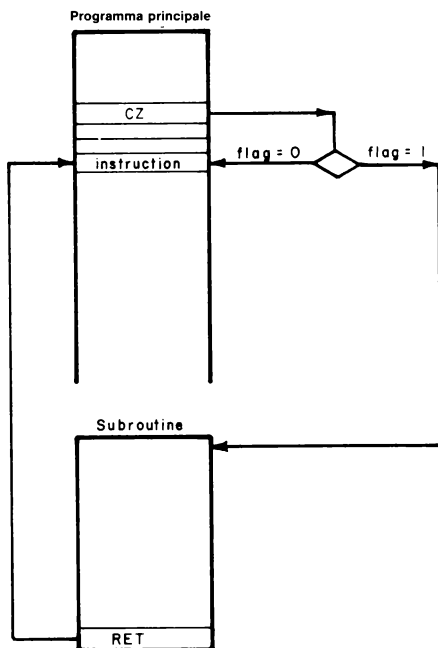
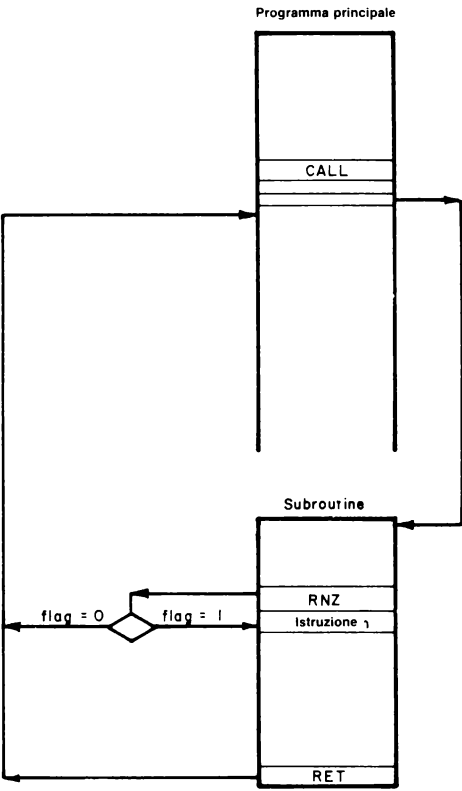
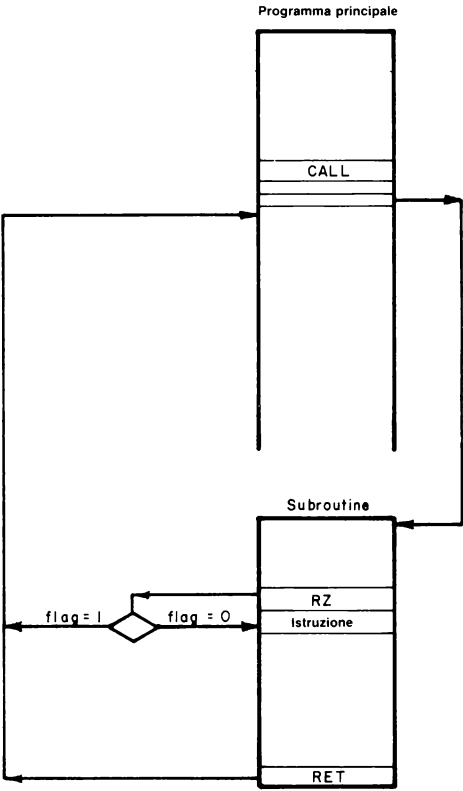
Rcondition (Rientro condizionato)
Se (CCC),
(PCL) ← ((SP))
(PCH) ← ((SP) + 1)
Se la condizione specificata è vera, vengono eseguite le azioni specificate nell'istruzione RET (vedi sopra); altrimenti, il controllo continua sequenzialmente.

Cicli: 1/3
Stati: 5/11
Indirizzamento: reg. indiretto
Flag: nessuno

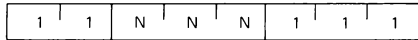
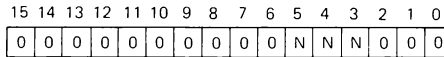
In un'istruzione di rientro condizionato, se la condizione è soddisfatta, avviene un rientro dalla subroutine; i contenuti del contatore di programma sullo stack vengono trasferiti al contatore di programma e l'esecuzione riprende all'istruzione immediatamente dopo l'istruzione di richiamo della subroutine.

Se la condizione non è soddisfatta, l'esecuzione del programma passa all'istruzione che segue immediatamente l'istruzione di rientro condizionato.

Le istruzioni condizionate, CZ, CNZ, RZ, e RNZ sono illustrate schematicamente nel diagramma che segue. Ricordate, Z significa che il flag di Zero dev'essere a livello logico 1 perchè avvenga un richiamo o un rientro; altrimenti, il controllo di programma passa all'istruzione successiva. NZ significa che il flag di Zero deve essere a livello logico 0 perchè avvenga un richiamo o un rientro; altrimenti, il controllo di programma passa all'istruzione successiva.

Programma principale
CALL
Subroutine
RZ
Istruzione
RET
flag = 1
flag = 0
Programma principale
CALL
Subroutine
RNZ
Istruzione
RET
flag = 0
flag = 1

RSTn

RST n (Ripristino)
((SP) − 1) ← (PCH)
((SP) − 2) ← (PCL)
(SP) ← (SP) − 2
PC) ← 8 * (NNN)

Gli otto bit di ordine più elevato dell'indirizzo dell'istruzione successiva vengono spostati nella locazione di memoria il cui indirizzo è uno meno del contenuto del registro SP. Gli otto bit di ordine inferiore dell'indirizzo dell'istruzione successiva vengono spostati nella locazione di memoria il cui indirizzo è due meno del contenuto del registro SP. Il contenuto del registro SP è decrementato di due. Il controllo è trasferito all'istruzione il cui indirizzo è otto volte il contenuto di NNN.

Contatore di programma dopo il ripristino

Citiamo il "*μCOM-8 Software Manual*": "Le istruzioni EI (Abilitare l'interruzione) e DI (Disabilitare l'interruzione) effettuano il controllo sull'accettazione di una richiesta di interruzione. Una volta che questo controllo è stabilito, il problema seguente da risolvere è come il dispositivo esterno indica al processore dove è posizionata la routine d'interruzione desiderata. L'8080A procede a questa identificazione permettendo al dispositivo di fornire un'istruzione quando viene riconosciuta l'interruzione. Benchè tutte le istruzioni 8080A possano essere specificate, solo due hanno valore pratico: un'istruzione di richiamo, CALL, e un'istruzione di ripristino, RST . . . . . Un'istruzione RST è un tipo specializzato di CALL. L'istruzione RST è un richiamo ad una delle otto locazioni in memoria specificata da un'espressione intera nella fascia di valori da 0 a 7 in codice ottale, indicata da N. Segue un elenco delle locazioni specificate dagli interi da 0 a 7.

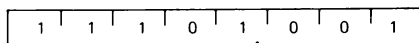
Valore di N	Locazione richiamata
0	HI = 000 e LO = 000
1	HI = 000 e LO = 010
2	HI = 000 e LO = 020
3	HI = 000 e LO = 030
4	HI = 000 e LO = 040
5	HI = 000 e LO = 050
6	HI = 000 e LO = 060
7	HI = 000 e LO = 070

Un'istruzione RST fa sì che il contatore di programma incrementato venga

inserito nello stack esattamente come accade con un'istruzione CALL. Carica poi il contatore di programma con HI = 000 e LO = ONO, dove N va da 0 a 7. RST 4 fa sì che il contatore di programma venga inserito nello stack e che HI = 000 e 000 e LO = 040 entrino nel contatore di programma."

"L'esecuzione del programma continua poi dal punto di ripristino. Se la routine di servizio dispositivo richiede più di otto byte per funzionare (come accade nella maggior parte dei casi), l'istruzione posta nel punto di ripristino deve determinare un salto alla subroutine di servizio interruzione. Dato che RST è un richiamo di subroutine specializzato, la subroutine di servizio interruzione *deve terminare con un'istruzione di rientro,* per ridare il controllo al programma interrotto riprendendo l'indirizzo di rientro."

"Dato che l'8080A ha soltanto otto istruzioni RST, qualunque livello aggiuntivo di interruzione deve essere implementato usando istruzioni CALL. Ciò significa che un'istruzione CALL addr deve essere fornita dal dispositivo d'interruzione, che è più difficile da implementare in hardware perchè CALL è un'istruzione a 3 byte. Comunque, una volta implementato, un richiamo diretto ad un routine è leggermente più veloce di un ripristino e di una successiva operazione di salto. Benchè questo non sia il fattore più importante, questa differenza in velocità di risposta dovrebbe essere tenuta in considerazione nel determinare come implementare le routine di servizio interruzione. Il primo beneficio che si ottiene usando la CALL è che un sistema di vettorizzazione ad "n" vie è fatto da hardware, eliminado il bisogno di software nella memoria di ordine inferiore (per l'elaborazione RST). Così quelle locazioni di memoria diventano libere per i programmi d'utente.

PCHL

L'istruzione PCHL fa sì che il contatore di programma venga caricato con i contenuti della coppia di registri HL. L'esecuzione del programma continua poi al punto designato dal contenuto di HL. In effetti, questa è un'istruzione di salto, ma, dal momento che sulla coppia di registri HL si può operare aritmeticamente, si possono implementare molti salti calcolati. La sequenza di istruzioni,

LXI H
<B2>
<B3>
PCHL

è identica in funzione a

JMP
<B2>
<B3>

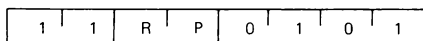
GRUPPO STACK, I/O, CONTROLLO MACCHINA

Questo gruppo di istruzioni esegue I/0, manipola lo stack, e altera i flag di controllo interni. Salvo diversa indicazione, *i flag di condizione non vengono coinvolti da nessuna istruzione di questo gruppo.*

PUSH rp e POP rp

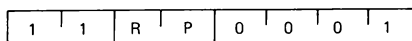
PUSH rp (Inserire nello stack)

((SP) − 1) ← (rh)
((SP) − 2) ← (rl)
(SP) ← (SP) − 2

Il contenuto del registro di ordine più elevato della coppia di registri rp è spostato nella locazione di memoria il cui indirizzo è uno meno del contenuto del registro SP. Il contenuto del registro di ordine inferiore della coppia di registri rp è spostato nella locazione di memoria il cui indirizzo è due meno del contenuto del registro SP. Il contenuto del registro SP è decrementato di 2. **Nota: La coppia di registri rp=SP può non essere specificata.**

Cicli: 3
Stati: 11
Indirizzamento: reg. indiretto
Flag: nessuno

POP rp (estrarre dallo stack)

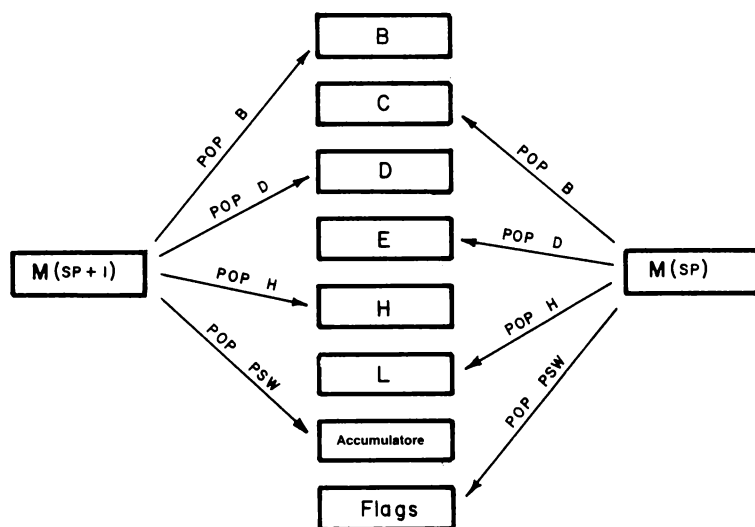
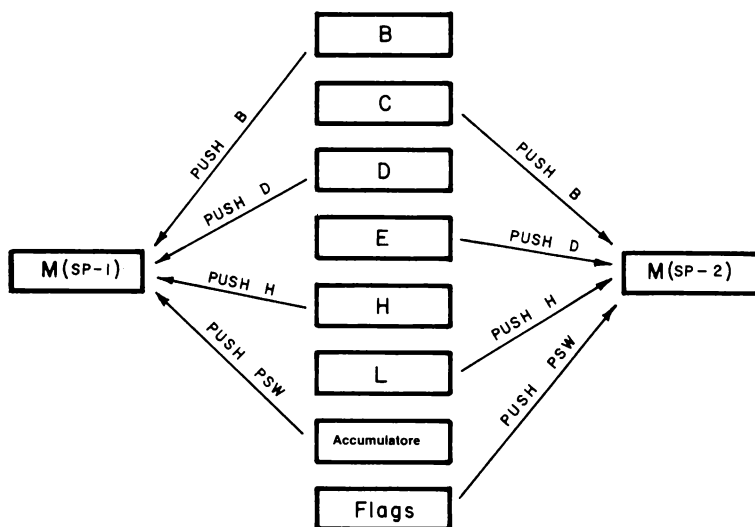
(rl) ← ((SP))
(rh) ← ((SP) + 1)
(SP) ← (SP) + 2

Il contenuto della locazione di memoria, il cui indirizzo è specificato dal contenuto del registro SP, è spostato nel registro di ordine inferiore della coppia di registri rp. Il contenuto della locazione di memoria, il cui indirizzo è uno più del contenuto del registro SP, è spostato nel registro di ordine più elevato della coppia di registri rp. Il contenuto del registro SP è incrementato di 2. **Nota: La coppia di registri rp=SP può non essere specificata.**

Cicli: 3
Stati: 10
Indirizzamento: reg. indiretto
Flag: nessuno

Citiamo il *"μCOM-8 Software Manual"*: "Due speciali istruzioni permettono al programmatore di mantenere e di reinserire in memoria i registri usando lo stack, PUSH e POP. PUSH rp fa sì che la coppia di registri specificata da RP venga posta nello stack. Lo stack è una parte speciale della memoria di lettura/scrittura designata dall'utente e usata come una memoria last-in-first-out (LIFO) per mezzo dello stack pointer a 16 bit. Un'operazione PUSH fa sì che lo stack pointer si decrementi di uno e memorizzi il registro più significativo (il registro HI) nella nuova locazione di memoria specificata dallo stack pointer. Lo stack pointer viene poi decrementato di nuovo e il registro meno significativo (il registro LO) viene poi memorizzato a quell'indirizzo. Per un'operazione POP, i dati nella locazione di memoria indirizzata dallo stack pointer vengono spostati nel registro meno significativo (il registro LO, che può essere C, E, o L); lo stack pointer è incrementato e i dati nella nuova locazione di memoria vengono caricati nel registro più significativo (il registro HI, che può essere B, D o H). Lo stack pointer viene poi incrementato di nuovo."

"Sia per le operazioni PUSH che POP, la coppia di registri, RP, può essere una delle tre coppie di registri BC, DE, o HL (identificate come B, D, e H, rispettivamente) o i contenuti del registro dei Flag e accumulatore, indicato da PSW (che sta per "program status word").

Le istruzioni PUSH e POP sono rappresentate schematicamente nella figura di pagina seguente. In questo diagramma, SP è la locazione originaria dello stack pointer, prima delle istruzioni PUSH o POP.

B
C
D
E
H
L
Accumulatore
Flags
M(SP-1)
M(SP-2)
PUSH B
PUSH D
PUSH H
PUSH PSW
B
C
D
E
H
L
Accumulatore
Flags
M(SP+1)
M(SP)
POP B
POP D
POP H
POP PSW

PSH psw e POP psw

PUSH psw (Inserire la parola di stato)

$((SP) - 1) \leftarrow (A)$
$((SP) - 2)_0 \leftarrow (CY),\ ((SP) - 2)_1 \leftarrow 1$
$((SP) - 2)_2 \leftarrow (P),\ ((SP) - 2)_3 \leftarrow 0$
$((SP) - 2)_4 \leftarrow (AC),\ ((SP) - 2)_5 \leftarrow 0$
$((SP) - 2)_6 \leftarrow (Z),\ ((SP) - 2)_7 \leftarrow (S)$
$(SP) \leftarrow (SP) - 2$

Il contenuto del registro A è spostato nella locazione di memoria il cui indirizzo è uno meno del registro SP. I contenuti dei flag vengono assemblati nella parola di stato e la parola è spostata nella locazione di memoria il cui indirizzo è due meno del contenuto del registro SP. Il contenuto del registro SP è decrementato di due.

Cicli: 3
Stati: 11
Indirizzamento: reg. indiretto
Flag: nessuno

POP psw (Estrarre la parola di stato)

$(CY) \leftarrow ((SP))_0$
$(P) \leftarrow ((SP))_2$
$(AC) \leftarrow ((SP))_4$
$(Z) \leftarrow ((SP))_6$
$(S) \leftarrow ((SP))_7$
$(A) \leftarrow ((SP) + 1)$
$(SP) \leftarrow (SP) + 2$

Il contenuto della locazione di memoria il cui indirizzo è specificato dal contenuto del registro SP viene usato per reinserire in memoria i flag di condizione. Il contenuto della locazione di memoria il cui indirizzo è uno più del contenuto del registro SP, è spostato nel registro A. Il contenuto del registro SP è incrementato di 2.

Cicli: 3
Stati: 10
Indirizzamento: reg. indiretto
Flag: Z,S,P,CY,AC

Le lettere, PSW, stanno per *"program status word"*; nel PSW vi sono i contenuti dell'accumulatore e dei cinque flag stato. Vi rimandiamo alla descrizione delle istruzioni PUSH rp e POP rp delle pagine precedenti. Il registro dei flag, F, è considerato il registro più significativo e l'accumulatore, A, è considerato il registro meno significativo. Il program status word è importante perchè mantiene lo stato macchina attuale così come è stato determinato dai cinque bit di flag. Quando PSW viene reinserito in memoria, le operazioni macchina possono riprendere dallo stato corretto, a prescindere da come la subroutine che ha interrotto coinvolge i flag.

PAROLA DEI FLAG

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z	0	AC	0	P	1	CY

Nel circuito integrato μCOM-8, che è funzionalmente identico al microprocessore 8080A, c'è un flag di stato in più, SUB. Nel registro dei flag, SUB occupa la posizione bit D_5. Inoltre, la posizione bit D_3 è a livello logico 1 e non a livello logico 0 (che è il caso del chip 8080A). Pensiamo che il flag SUB sia una caratteristica utile dei microprocessori tipo 8080A, e speriamo che venga incorporato nelle versioni future del chip dai fabbricanti, quali Texas Instruments, National Semiconductor, Intel, Siemens, ecc.

Un esempio di come opera lo stack viene dato nella figura di pagina seguente. La sezione di programma usata è,

Programma principale	*Subroutine*
LXI-SP	PUSH B
303	PUSH D
003	PUSH H
CALL	PUSH PSW
< B2 >	
< B3 >	

Originariamente lo stack pointer era posizionato a HI = 003 e LO = 303.
Dopo l'istruzione CALL, i due contatori di programma vengono inseriti sullo stack e lo stack pointer si sposta su HI = 003 e LO = 301. Notate che il byte del contatore di programma HI va sullo stack per primo ma se ne allontana per ultimo. Una successione di quattro istruzioni PUSH carica lo stack con i contenuti dei sei registri universali, dell'accumulatore, e del registro dei flag. Dopo tutto ciò, la locazione dello stack pointer (SP) è HI = 003 e LO = 271, la posizione occupata in alto sullo stack.

Una volta che la subroutine è stata eseguita, vi è il problema di spostare i contenuti dello stack e di rimetterli nel microprocessore 8080A. La sezione di programma, posizionata alla fine della subroutine, che assolve questo compito, è

```
POP PSW
POP H
POP D
POP B
RET
```

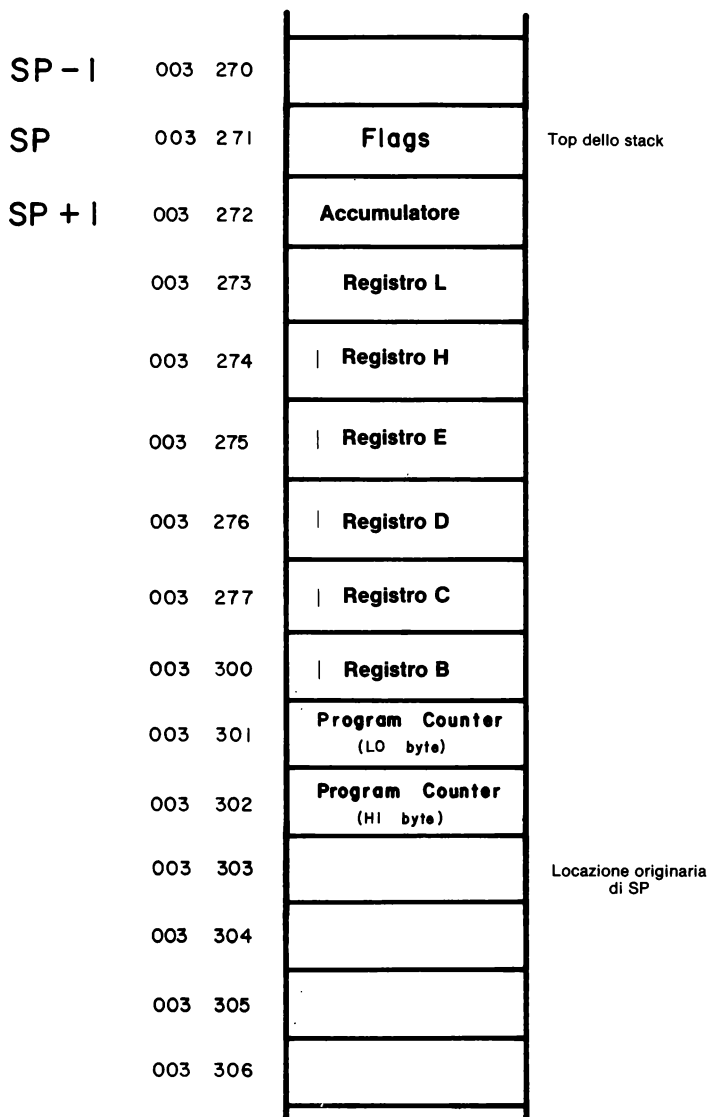
In ogni caso, il byte LO si allontana dallo stack per primo. Ricordate che nelle istruzioni a tre byte, il byte LO è sempre il secondo byte dell'istruzione. Quindi, il chip 8080A è ben preciso nella sua gestione di parole di 16 bit. Dopo che i contenuti dello stack sono stati estratti, lo stack pointer riprende la sua locazione originale ad HI = 003 e LO = 303.

I registri possono essere inseriti e prelevati in qualunque ordine. Comunque, il contatore di programma è quasi sempre inserito per primo e prelevato per ultimo. Dovete stare attenti a prelevare i registri nell'ordine inverso a quello nel quale li avete inseriti. Ad esempio, con la configurazione dello stack mostrata nella pagina seguente, se eseguiste la seguente sezione di programma alla fine della subroutine,

```
POP PSW
POP B
POP H
POP D
RET
```

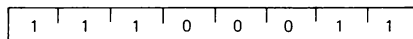
avreste dei problemi nell'eseguire il programma principale. I contenuti originari del registro non ritornerebbero nelle loro locazioni originarie. Il chip tenterebbe di eseguire il programma, ma vi sarebbero poche possibilità di ottenere un risultato utile.

Se non avete bisogno di inserire il registro sullo stack durante un richiamo della subroutine, non fatelo.Memorizzate solo quell'informazione di cui il chip 8080A ha bisogno quando riprende il programma principale.

	Indirizzo	Contenuto	
SP − 1	003 270		
SP	003 271	Flags	Top dello stack
SP + 1	003 272	Accumulatore	
	003 273	Registro L	
	003 274	Registro H	
	003 275	Registro E	
	003 276	Registro D	
	003 277	Registro C	
	003 300	Registro B	
	003 301	Program Counter (LO byte)	
	003 302	Program Counter (HI byte)	
	003 303		Locazione originaria di SP
	003 304		
	003 305		
	003 306		

Lo stack

XTHL

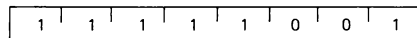
XTHL (Scambiare la parte superiore dello stack con H ed L)
(L) ←→ ((SP))
(H) ←→ ((SP) + 1)
Il contenuto del registro L è scambiato con il contenuto della locazione di memoria il cui indirizzo è specificato dal contenuto del registro SP. Il contenuto del registro H è scambiato con il contenuto della locazione di memoria il cui indirizzo è uno più del contenuto del registro SP.

1	1	1	0	0	0	1	1

Cicli: 5
Stati: 18
Indirizzamento: reg. indiretto
Flag: nessuno

L'istruzione XTHL viene usata per scambiare i contenuti della coppia di registri HL con la coppia superiore di elementi sullo stack. I contenuti della locazione superiore, quelli indirizzati dallo stack pointer SP, vengono scambiati con i contenuti del registro L. Lo stack pointer è incrementato, e i contenuti di memoria indirizzati da questo nuovo valore di SP, vengono scambiati con i contenuti del registro H.

SPHL

SPHL (Spostare HL a SP)
(SP) ← (H) (L)
I contenuti dei registri H ed L (16 bit) sono spostati nel registro SP.

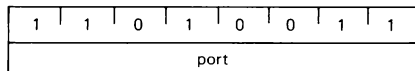
1	1	1	1	1	0	0	1

Cicli: 1
Stati: 5
Indirizzamento: registro
Flag: nessuno

L'istruzione SPHL viene usata per caricare il registro stack pointer con i contenuti della coppia di registri H, L. I contenuti di L vengono posti negli otto bit LO dello stack pointer, e i contenuti di H sono posti negli otto bit H dello stack pointer. Come rilevato nel Manuale NEC Microcomputers, Inc: "L'istruzione SPHL può essere usata per caricare lo stack pointer con un valore che è stato elaborato usando le operazioni aritmetiche su due registri disponibili con la coppia di registri HL. Questo andrebbe sempre fatto con attenzione, dato che è facile perdere la traccia di dove lo stack pointer sta puntando, con conseguente perdita del contenuto dello stack."

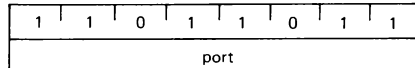
OUT port

L'istruzione OUT port sposta i contenuti a 8 bit dell'accumulatore nella porta di uscita specificata dal secondo byte dell'istruzione.

Possono essere selezionate 256 porte di uscita. Durante il terzo ciclo macchina dell'istruzione, il codice dispositivo appare sul bus d'indirizzo, viene generato un impulso di controllo $\overline{\text{OUT}}$, e i contenuti dell'accumulatore appaiono sul bus di dati bidirezionale esterno.

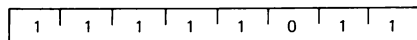
OUT port (Uscita)
(data) ← (A)
Il contenuto del registro A è posto sul bus di dati bidirezionale a 8 bit per la trasmissione alla porta specificata.

Cicli: 3
Stati: 10
Indirizzamento: diretto
Flag: nessuno

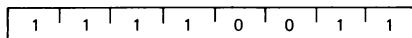
IN port

IN port (Ingresso)
(A) ← (data)
I dati posti sul bus di dati bidirezionale a otto bit dalla porta specificata, sono spostati nel registro A.

Cicli: 3
Stati: 10
Indirizzamento: diretto
Flag: nessuno

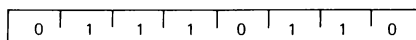
L'istruzione IN port permette al chip 8080A di leggere i dati presenti alla porta d'ingresso data dal secondo byte dell'istruzione. Possono essere indirizzate 256 porte d'ingresso. Durante il terzo ciclo macchina dell'istruzione, il codice dispositivo per il dispositivo di ingresso appare sul bus d'indirizzo, sul bus di controllo appare un segnale di controllo $\overline{\text{IN}}$, e e l'informazione che appare sul bus di dati bidirezionale appare anche nell'accumulatore.

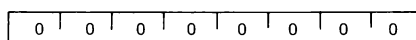
EI e DI

EI (Abilitare le interruzioni)
Il sistema di interruzione è abilitato
dopo l'esecuzione dell'istruzione successiva.

DI (Disabilitare le interruzioni)
Il sistema di interruzione è disabilitato
immediatamente dopo l'esecuzione dell'istruzione DI.

EI:
Cicli: 1
Stati: 4
Flag: nessuno

DI:
Cicli:1
Stati: 4
Flag: nessuno

Citiamo il *μCOM-8 Software Manual* della NEC Microcomputers, Inc.: "Il fatto che l'8080A risponda o meno ad una richiesta d'interruzione è determinato dallo stato di un flip-flop interno di interrupt, INTE. Quando questo flip-flop è posizionato su uno, il processore risponde alle interruzioni. Quando viene resettato a zero, il processore ignora le richieste d'interruzione. Il flip-flop INTE è coinvolto sia dal controllo di programma che dall'operazione di sistema. Le operazioni di sistema che coinvolgono INTE sono un reset del sistema e il riconoscimento di un'interruzione. Entrambe le operazioni azzerano INTE e disabilitano così l'abilitazione all'interruzione.
Se devono essere rilevate altre interruzioni dopo un reset o un riconoscimento d'interruzione, il programma deve riabilitare il flip-flop. Due istruzioni, EI (Enable Interrupt) e DI (Disable Interrupt), forniscono il controllo programmato del flip-flop INTE. L'istruzione EI posiziona il flip-flop INTE su uno, abilitando l'interruzione, mentre l'istruzione DI azzera il flip-flop INTE, disabilitando l'interruzione. Lo stesso accade se si desidera che una sezione del programma venga eseguita alla massima velocità e senza possibilità d'interruzione, l'istruzione DI può essere usata per disabilitare le interruzioni per quella sezione di codice. Dopo che la sezione è completa, EI riabilita l'interruzione. Dato che il riconoscimento di una richiesta d'interruzione resetta il flip-flop INTE su zero, in qualunque routine, la prima istruzione a servire le interruzioni dovrebbe essere una EI. (Questo presuppone che il riconoscimento dell'interruzione resetti la richiesta d'interruzione stessa. Ciò deve essere fatto per evitare che il processore 8080A abbia un arresto imprevisto). Bisognerebbe fare un'eccezione quando viene servito il dispositivo di I/O più veloce. Per evitare che questa unità di I/O venga disturbata, il flip-flop INTE dovrebbe essere abilitato alla fine della routine."

HLT

L'istruzione HLT fa sì che il processore sospenda l'operazione finchè il chip 8080A riceve un segnale RESET o riceve un segnale di richiesta d'interruzione(INT). Il processore accetta la richiesta INT a preescindere dalla condizione del flip-flop interno di interrupt. Dopo che l'interruzione è stata elaborata, l'esecuzione dell'istruzione continua alla posizione successiva dopo il comando di Halt.

NOP

L'istruzione NOP non fa assolutamente niente all'infuori di occupare una locazione in memoria e prelevare quattro stati durante l'esecuzione del programma. Viene usata per la messa a punto di programmi, nei quali le istruzioni NOP in più vengono poste in un programma per una modifica successiva. Quando in un programma vengono fatte delle cancellazioni, le istruzioni NOP dovrebbero essere inserite al loro posto.

* * *

Con l'aiuto del materiale dell'"*Intel 8080 Microcomputers Systems User's Manual*" e il "*µCOM-8 Software Manual*" della NEC Microcomputers, abbiamo fornito una descrizione minuziosa delle singole istruzioni del chip 8080A. Siamo grati sia alla Intel Corporation che alla NEC Microcomputers, Inc. per il gentile permesso concessoci di usare le informazioni dei loro manuali. Se lavorate seriamente sul chip 8080A, dovreste possedere entrambi i manuali.

APPENDICE 3

Sommario del Set di Istruzione dell'8080 (Intel Corp.)

Mnemonic	Description	Instruction Code [1] D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock [2] Cycles
MOV [illegible]	Move register to register	0	1		D	D	S	S		5
MOV M [illegible]	Move register to memory	0			1	[illegible]	S			7
MOV r M	Move memory to register				0	D				7
HLT	Halt				1	0				7
MVI [illegible]	Move immediate register			1	0	D				7
[illegible]	Move immediate memory			1	1	0				10
	Increment register			0	0	[illegible]				5
	Decrement register			0	D					5
	Increment memory			1	1					1[illegible]
	Decrement memory									
	Add register to A									
	[illegible]dd register to A w[illegible]									
	[illegible]ract register									
AN[illegible]		1	0							
XRA [illegible]	[illegible]ster with A	1	0	1	[illegible]					
ORA [illegible]	[illegible]gister with A	1	0	1	1	0	[illegible]			4
CMP r	Compare register with A	1	0	1	1	1	S	S	S	
ADD M	Add memory to A	1	0	0	0	0	1	1	0	
ADC M	Add memory to A with carry	1	0	0	0	1	1	1	0	[illegible]
SUB M	Subtract memory from A	1	0	0	1	0	1	1	0	7
SBB M	Subtract memory from A with borrow	1	0	0	1	1	1	1	0	7
ANA M	And memory with A	1	0	1	0	0	1	1	0	7
XRA M	Exclusive Or memory with A	1	0	1	0	1	1	1	0	[illegible]
ORA M	Or memory with A	1	0	1	1	0	1	1	0	7
CMP M	Compare memory with A	1	0	1	1	1	1	1	0	7
ADI	Add immediate to A	1	1	0	0	0	1	1	0	7
ACI	Add immediate to A with carry	1	1	0	0	1	1	1	0	7
SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	7
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	7
ANI	And immediate with A	1	1	1	0	0	1	1	0	7
XRI	Exclusive Or immediate with A	1	1	1	0	1	1	1	0	
ORI	Or immediate with A	1	1	1	1	0	1	1	0	
CPI	Compare immediate with A	1	1	1	1	1	[illegible]	1	0	
RLC	Rotate A left	0	0	0	0	0	1			
RRC	Rotate A right	0	0	0	0	1				
RAL	Rotate A left through carry	0	0	0						
RAR	Rotate A right through c[illegible]	0	0						1	
JMP	J[illegible]							1	1	
JC	J[illegible] carry							1	0	10
JNC	Jmp on no carry							1	0	10
JZ	Ju[illegible]		1					1	0	10
JNZ	Ju[illegible]		[illegible]				0	1	0	10
JP	Jump o[illegible]	1	1			0	0	1	0	10
JM	Jump on minus	1				1	0	1	0	10
JPE	Jump on parity even	1	1	[illegible]	0	1	0	1	0	10
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	10
CALL	Call unconditional	1	1	0	0	1	1	0	1	17
CC	Call on carry	1	1	0	1	1	1	0	0	11 17
CNC	Call on no carry	1	1	0	1	0	1	0	0	11 17
CZ	Call on zero	1	1	0	0	1	1	0	0	11 17
CNZ	Call on no zero	1	1	0	0	0	1	0	0	11 17
CP	Call on positive	1	1	1	1	0	1	0	0	11 17
CM	Call on minus	1	1	1	1	1	1	0	0	11 17
CPE	Call on parity even	1	1	1	0	1	1	0	0	11 17
CPO	Call on parity odd	1	1	1	0	0	1	0	0	11 17
RET	Return	1	1	0	0	1	0	0	1	10
RC	Return on carry	1	1	0	1	1	0	0	0	5/11
RNC	Return on no carry	1	1	0	1	0	0	0	0	5/11

Mnemonic	Description	Instruction Code [1] D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock [2] Cycles
RZ	Return on zero	1	1	0	0	1	0	0	0	5/11
RNZ	Return on no zero	1	1	0	0	0	0	0	0	5/11
RP	Return on positive	1	1	1	1	0	0	0	0	5/11
RM	Return on minus	1	1	1	1	1	0	0	0	5/11
RPE	Return on parity even	1	1	1	0	1	0	0	0	5/11
RPO	Return on parity odd	1	1	1	0	0	0	0	0	5/11
RST	Restart	1	1	A	A	A	1	1	1	11
IN	Input	1	1	0	1	1	0	1	1	10
OUT	Output	1	1	0	1	0	0	1	1	10
LXI B	Load immediate register Pair B & C	0	0	0	0	0	0	0	1	10
LXI D	Load immediate register Pair D & E	0	0	0	1	0	0	0	1	10
LXI H	Load immediate register Pair H & L	0	0	1	0	0	0	0	1	10
LXI SP	Load immediate stack pointer	0	0	1	1	0	0	0	1	10
PUSH B	Push register Pair B & C on stack	1	1	0	0	0	1	0	1	11
PUSH D	Push register Pair D & E on stack	1	1	0	1	0	1	0	1	11
PUSH H	Push register Pair H & L on stack	1	1	1	0	0	1	0	1	11
PUSH PSW	Push A and Flags on stack	1	1	1	1	0		0	1	11
POP B	Pop register pair B & C off stack	1	1	0	0	0	0	0	1	10
POP D	Pop register pair D & E off stack	1	1	0	1	0	0	0	1	10
POP H	Pop register pair H & L off stack	1	1	1	0	0	0	0	1	10
POP PSW	Pop A and Flags off stack	1	1		1	0	0	0	1	10
STA	Store A direct	0	0	1	1	0	0	1	0	13
[illegible]DA	Load A direct	0	0	1	1	1	0	1	0	13
[illegible]HG	Exchange D & E H & L Registers	1	1	1	0	1	0	1	1	4
XTHL	Exchange top of stack H & L	1	1	1	0	0	0	1	1	18
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1	5
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	5
DAD B	Add B & C to H & L	0	0	0	0	1	0	0	1	10
DAD D	Add D & E to H & L	0	0	0	1	1	0	0	1	10
DAD H	Add H & L to H & L	0	0	1	0	1	0	0	1	10
DAD SP	Add stack pointer to H & L	0	0	1	1	1	0	0	1	10
STAX B	Store A indirect	0	0	0	0	0	0	1	0	7
STAX D	Store A indirect	0	0	0	1	0	0	1	0	7
LDAX B	Load A indirect	0	0	0	0	1	0	1	0	7
LDAX D	Load A indirect	0	0	0	1	1	0	1	0	7
INX B	Increment B & C registers	0	0	0	0	0	0	1	1	5
INX D	Increment D & E registers	0	0	0	1	0	0	1	1	5
INX H	Increment H & L registers	0	0	1	0	0	0	1	1	5
INX SP	Increment stack pointer	0	0	1	1	0	0	1	1	5
DCX B	Decrement B & C	0	0	0	0	1	0	1	1	5
DCX D	Decrement D & E	0	0	0	1	1	0	1	1	5
DCX H	Decrement H & L	0	0	1	0	1	0	1	1	5
DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1	5
CMA	Complement A	0	0	1	0	1	1	1	1	4
STC	Set carry	0	0	1	1	0	1	1	1	4
CMC	Complement carry	0	0	1	1	1	1	1	1	4
DAA	Decimal adjust A	0	0	1	0	0	1	1	1	4
SHLD	Store H & L direct	0	0	1	0	0	0	1	0	16
LHLD	Load H & L direct	0	0	1	0	1	0	1	0	16
EI	Enable Interrupts	1	1	1	1	1	0	1	1	4
DI	Disable interrupt	1	1	1	1	0	0	1	1	4
NOP	No operation	0	0	0	0	0	0	0	0	4

NOTE: 1. DDD o SSS - 000 B - 001 C - 010 D - 011 E - 100 H - 101 L - 110 Memory - 111 A

2. Due possibili tempi di ciclo, (5/11) indicano che il ciclo dell'istruzione dipende dai flag di condizione

L. 19.000
(17.924)

Edizione italiana del "the BUGBOOK III"

JACKSON
ITALIANA
EDITRICE

P.R. RONY
D.G. LARSEN
J.A. TITUS

10